2

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

ON-ORBIT ANNEALING OF GALLIUM ARSENIDE
SOLAR CELL ARRAYS

by

Robert Stefan Sommers

June 1989

Thesis Advisor	Sherif Michael

approved for public release; distribution is unlimited

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

3. DISTRIBUTION / AVAILABILITY OF REPORT
Approved for public release; distribution is unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
Naval Postgraduate School

6b. OFFICE SYMBOL (If applicable)
39

7a. NAME OF MONITORING ORGANIZATION
Naval Postgraduate School

6c. ADDRESS (City, State, and ZIP Code)
Monterey, CA 93943-5000

7b. ADDRESS (City, State, and ZIP Code)
Monterey, CA 93943-5000

8a. NAME OF FUNDING / SPONSORING ORGANIZATION

8b. OFFICE SYMBOL (If applicable)

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS (City, State, and ZIP Code)

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
ON-ORBIT ANNEALING OF GALLIUM ARSENIDE SOLAR CELL ARRAYS

12. PERSONAL AUTHOR(S)
Sommers, Robert S.

13a. TYPE OF REPORT
Master's Thesis

13b. TIME COVERED
FROM ________ TO ______

14. DATE OF REPORT (Year, Month, Day)
1989 June

15. PAGE COUNT
127

16. SUPPLEMENTARY NOTATION The views expressed in this thesis are thoses of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Solar Cell; GaAs Cell; Digital Design; IV Curves; Solar Cell Annealing

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

A complete experimental package is developed for the acquisition of current-voltage curves and the annealing of Gallium Arsenide solar cells and arrays of cells. This package is designed to be placed aboard a wide variety of satellite buses and to operate with little interface to the satellite system.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL
Sherif Michael

22b. TELEPHONE (Include Area Code)
(408) 646-2252

22c. OFFICE SYMBOL
62Mi

Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

On-Orbit Annealing of Gallium Arsenide Solar Cell Arrays

by

Robert Stefan Sommers
Lieutenant, United States Navy
B.A., Trinity University, 1976

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
1989

Author: Robert Stefan Sommers
Robert Stefan Sommers

Approved by: Sherif Michael
Sherif Michael, Thesis Advisor

Herschel Loomis, Second Reader

M. A. Morgan
Michael A. Morgan, Chairman (acting)
Department of Electrical and Computer Engineering

G E Schacher
Gordon E. Schacher,
Dean of Science and Engineering

ABSTRACT

A complete experimental package is developed for the acquisition of current-voltage curves and the annealing of Gallium Arsenide solar cells and arrays of cells. This package is designed to be placed aboard a wide variety of satellite buses and to operate with little interface to the satellite system.

TABLE OF CONTENTS

I. INTRODUCTION

Solar energy has been and will continue to be the primary source of power on nearly all earth-orbiting spacecraft. Photovoltaic devices, primarily solar cells, have proven to be a simple and reliable mechanism for the conversion of the sun's radiant energy into usable electrical energy.

Typical solar array subsystems are designed to provide adequate power to the spacecraft through the end-of-life (EOL) which, depending on design requirements, may be as long as ten years. The space environment is particularly harsh on orbiting satellites and, depending on the altitude of the orbit, can destroy solar cell arrays in less than a year. Even in low-radiation orbits, geomagnetically trapped protons, electrons and solar flare protons damage the solar cell arrays resulting in decreased efficiencies with an overall reduction in available output power.

To meet the EOL power requirements, solar arrays are designed to provide more power than necessary to account for radiation degradation and other degrading effects. Thus, beginning-of-life (BOL) power is normally more than needed and usually some percentage of it is wasted. New requirements for higher power systems make this approach very expensive in both launch weight and component cost [Fig. 1: Ref. 1]. Attempts to minimize the quantity of solar cells required at BOL have been primarily concentrated in the areas of new cover

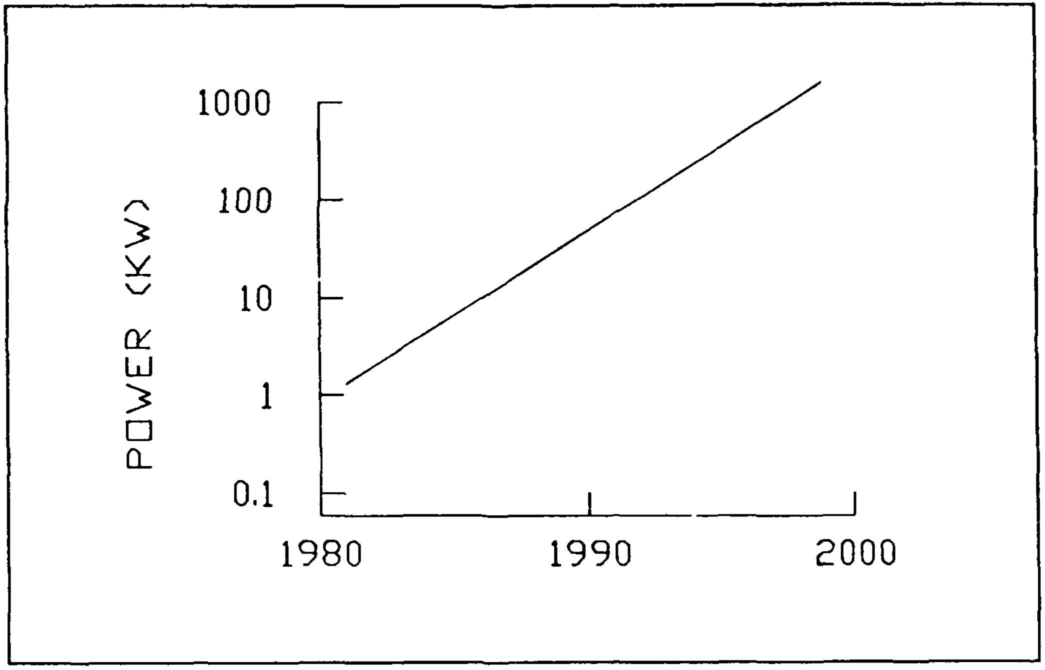


Figure 1 Power Requirements For Military Spacecraft 1980-2000 [adapted from Ref. 1]

materials, radiation hardening of the cells, development of more efficient cells and, to some extent, the thermal annealing of cell arrays.

The process of thermal annealing Si cells (realigning displaced molecules within the crystalline structure of the cell by raising it to a high temperature) has been considered impractical on-orbit, due to the excessive temperatures involved. GaAs cells, on the other hand, anneal at much lower temperatures. By using the technique of foiwaid-biased current annealing, the required temperature drops to such a levei that this annealing can easily be achieved on-orbit.

A. PROBLEM DEFINITION AND GOALS

As the power required for satellite systems increases the size of the solar arrays will increase as well. Replacing currently used Si cells with GaAs cells would provide a significant reduction in the number of cells required. Additionally, incorporating forward-biased annealing into the array subsystem would reduce the required number of cells to an even greater extent as well as extend the life of the satellite.

Forward-biased current annealing has never been performed on-orbit. In fact, the ability to monitor a solar cell's performance on-orbit has been rather poor due to the large and heavy equipment required to do an adequate job.

Research at the Naval Postgraduate School (NPS) has provided two theories/devices which require testing on-orbit. The first involves the use of a novel circuit which acquires and records cell characteristic data (current/voltage curves) on individual solar cells. The second involves a circuit which steers annealing currents to forward-bias a solar cell. Both circuits have been constructed and thoroughly tested in the laboratory with highly successful results. This thesis describes in detail the design and construction of these circuits and their integration into a single, computer controlled, experimental package. This package, the NPS Solar Experiment, has been constructed in prototype and operates as presented.

The data acquired by testing this experiment in space will give additional insight into the degrading effects of the space environment as well as providing proof that forward-biased current annealing is both possible and beneficial on-orbit.

B. TECHNICAL BACKGROUND

1. DETERMINATION OF CELL STATUS

In order to determine a solar cell's performance (or performance degradation) it is necessary to develop a benchmark. This benchmark is the cell current-voltage (IV) curve [Figure 2]. From the IV curve, the cell's maximum power point as well as open circuit voltage (Voc) and short circuit current (Isc) can be determined. The effects of the environment in which a cell is operating can mainly be determined from these parameters.

The requirement for testing solar cells on-orbit led to the development of a novel circuit which is fast, accurate, consumes very little power and is extremely portable. The circuitry is based on the fact that a bipolar junction transistor models an ideal current source/sink. Therefore, this can be used to control the output current of a solar cell starting at zero current (Voc) while acquiring the corresponding voltages and currents up to and including zero voltage (Isc). Detailed discussion of this circuit is provided Chapter V.

2. SILICON VERSUS GALLIUM ARSENIDE CELLS

Silicon (Si) solar cells are considered the main power source for modern day satellite systems. They have proven to be durable, reliable and have experienced a significant cost reduction over the years. However, as power requirements increase, the required size of Si arrays will become prohibitive. GaAs cell arrays

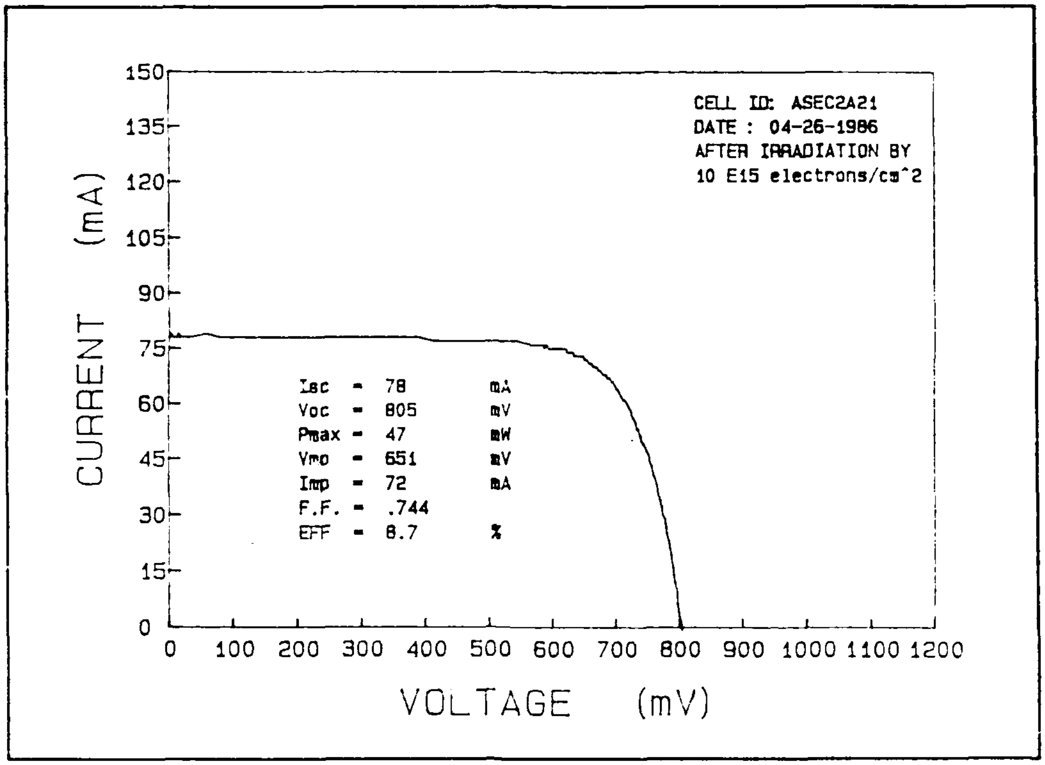


Figure 2 Typical IV Curve for Heavily
Irradiated 2x2 cm GaAs Solar Cell

are an attractive alternative due to their power generating capabilities (higher efficiencies), and their greater radiation resistance to nearly all radiation sources [Ref. 2]. The four-fold cost increase for using GaAs in place of Si will become less important as the use of Si becomes prohibitive. In the mean time, as more GaAs cells are utilized in spacecraft design, the cost of these new cells will become more comparable to Si cells.

Standard silicon cells (2x2 cm) produce output voltage in the range of 500 to 550 millivolts at a current of 120 milliamps when illuminated in air mass zero (AM0). Gallium arsenide cells, on the other hand, produce an output voltage in the range of 950 to 1000 millivolts at a typical current of 130 milliamps. It can be seen immediately that the power generated by the GaAs cell is significantly higher than that of the Si cell. Use of these cells with no further consideration would dramatically reduce the amount of cells required at BOL.

GaAs cells have an additional advantage over Si cells that is extremely important. GaAs cells can be annealed at temperatures significantly lower than that of their Si counterparts. Such lower temperatures may be more practical to achieve on a solar array surface in space.

Radiation-degraded silicon solar cells experience significant thermal annealing at temperatures ranging from 450 to 550° C [Ref. 3]. These temperatures would be extremely difficult to generate on-orbit and is prohibitive due to the damage that would be incurred to the contacts and cover materials.

GaAs cells, on the other hand, thermally anneal in the vicinity of 200° C. Actual stages of recovery have been reported to occur at 150-200, 250-300 and 400-500° C [Ref. 4]. Nonetheless, 200° C is still a difficult barrier to reach on-orbit. Research at the Naval Postgraduate School has shown that cells achieve significant annealing by forward-biasing three amperes of current through a 2x2 cm GaAs cell and maintaining the cell's temperature at 90 to 100° C for 24 hours. Test cells that have been irradiated with 1-Mev electrons to fluence levels of 3 x 10^{15} electrons/cm^2 and have lost 40% of their original efficiency recover approximately 33% of that lost efficiency upon annealing. [Figures 3,4: Ref. 5]

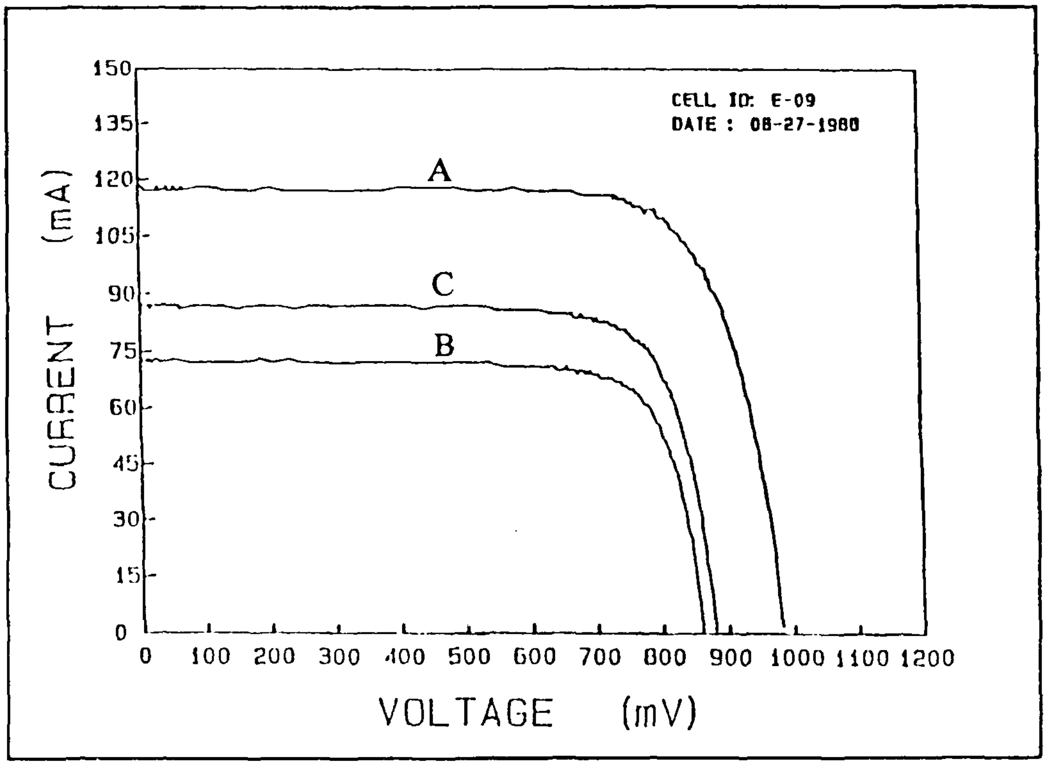


Figure 3 IV Curve of GaAs Cell E-09 [Ref. 5]

A. Cell prior to irradiation
B. Cell after 3 x 10^{15} 1-Mev electrons
C. Cell after annealing

The promising aspect of these results is that the required temperature is available on-orbit and the current required will not place a major burden on the satellite power supply. As can easily be shown, it will require 28 cells to generate enough power to anneal one other cell. A segmented array can be envisioned that, under computer control, tests and anneals cells and arrays as necessary.

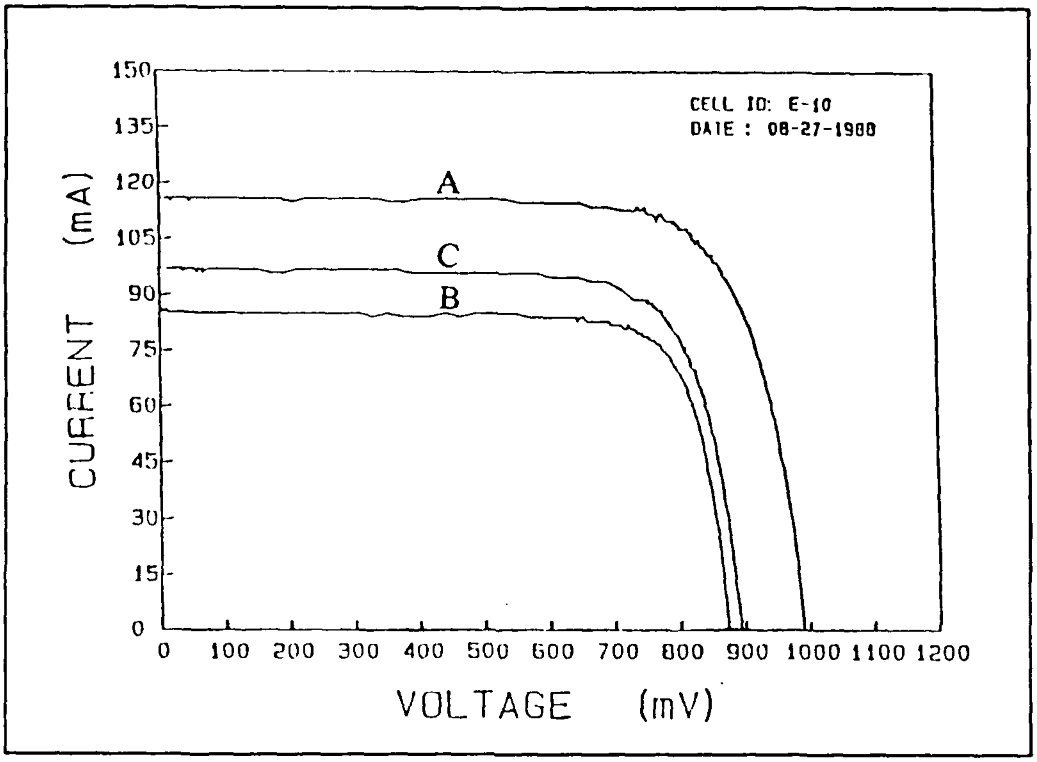


Figure 4 IV Curve of GaAs Cell E-10 [Ref. 5]

A. Cell prior to irradiation
B. Cell after 3 x 10^{15} 1-Mev electrons
C. Cell after annealing

3. ANNEALING OF CELLS

Once the determination has been made to anneal a cell, the cell must be removed from active power production to the satellite, the IV curve circuitry must be isolated and current must be forward-biased into the cell. This job is easily handled by allowing a compyter to secure power buses and shunt annealing power to the cell or array of cells. Three amperes per cell is provided to the array and is maintained for 24 hours. Since 24 hours is considered optimum annealing duration, the cell is not checked again until the next IV curve cycle.

4. NPS SOLAR EXPERIMENT

The NPS Solar Experiment is designed to be generally autonomous. The only requirements from the satellite that will incorporate this experiment are:

1. A 12x17x18 cm space for mounting the experiment's chassis.

2. An 8x8 cm external satellite surface pointing towards the sun on which to mount the test cells.

3. A 0.9 Kg weight allowance.

4. Five ampere, + 28 volt and one ampere, -28 volt power sources.

5. An interrupt line from the satellite main computer to the experiment's controller which alerts the experiment controller when to download it's data.

6. A serial port receive line operating at 9600 baud, no start bits, eight data bits and one stop bit.

7. Prior knowledge of the satellite's orbital parameters.

The seventh requirement is of particular interest. In order to take full advantage of the experiment's capabilities, the altitude of the satellite bus must be known. If the satellite is orbiting in low-earth orbit (LEO) (e.g., a 'Get-Away Special' (GAS) satellite), the cells will not degrade enough in a period of two to three years to require annealing. With the experiment on a LEO satellite, the IV curves could still be obtained for each cell. The annealing function would be secured. (This IV curve acquisition function alone has only been done once in space and with limited results [Ref. 6].) The ideal bus for taking full advantage of this experiment would transit the Van Allen belts frequently. Due to the high radiation levels in these belts, more severe cell degradation would occur. Thus, the on-orbit annealing portion of the experiment would produce more useful results.

The design of the operational prototype is expected to be the design of the flight-ready experimental package with only slight modifications to match the interface documents provided by the owners of the host satellites. (Every satellite interface will be slightly different.) In it's current form, sixteen 2x2 cm GaAs solar cells would be arranged in a square which measures 8x8 cm with each cell connected independently to the experiment controller. (The cells will perform no power generation for the satellite bus.) IV curves would be acquired on each of the cells once every ten days and the data stored in RAM. The satellite's main computer would then signal the experiment to serially transmit the IV curve data to its memory. Once the satellite has received the data from the Solar Experiment, it would transmit it to the ground. The ground station program of Appendix G, converts the data to floating point decimal. This data would then be plotted to generate the IV curves of interest.

Every 180 days the IV curve cycle would stop and the anneal cycle would begin. Each cell is annealed for 24 hours. At the end of this 16 day period, all clocks would be returned to zero. Ten days later, the first IV curves on the newly annealed cells would become available.

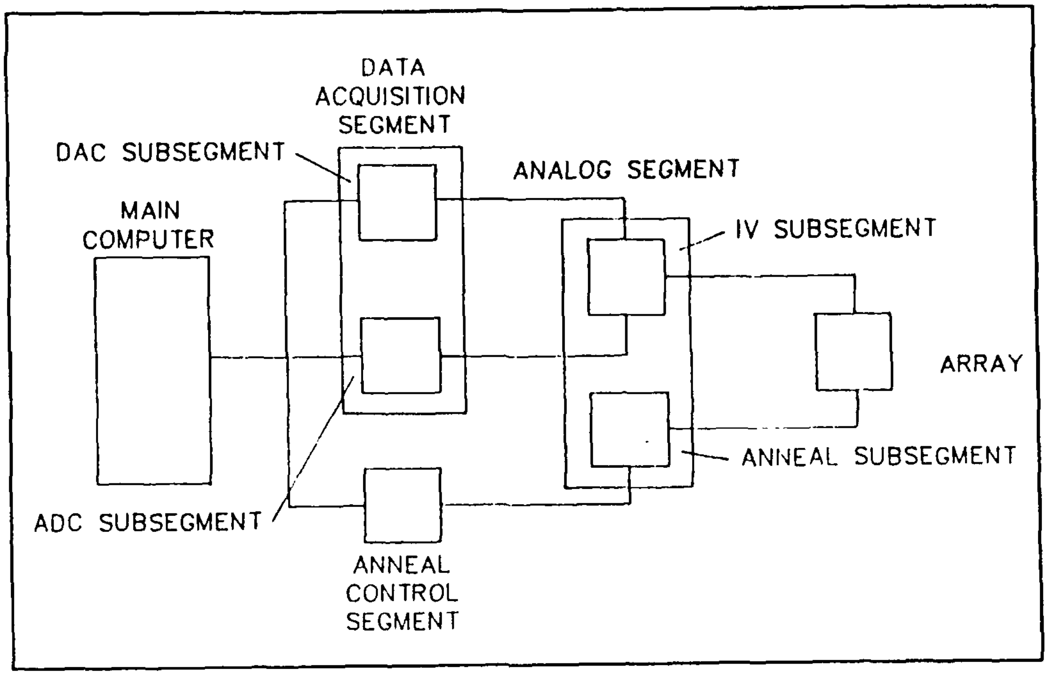


Figure 5 Block Diagram for NPS Solar Cell Experiment

The block diagram of Figure 5 shows the overall layout of the experiment. The main computer supplies the DAC (digital to analog) segment with a sequential count from zero to 255. The DAC segment converts this digital signal to analog voltage which is applied to the base of the IV curve transistor. This voltage 'stair step' incrementally increases the amount of current flowing through the solar cell which is interfaced to the analog segment. The ADC (analog to digital) segment measures the voltages in the analog segment during this current sweep and converts them back to digital so they can be stored in the main computer's RAM. (These voltages represent different points along the IV curve.) This information is then transmitted at the command of the satellite's controller. The anneal segment responds to the main computer by either turning the anneal transistors in the analog segment on or off, thereby providing or securing annealing current to a cell. No data is acquired during the anneal cycle.

The actual hardware designs for the experiment are divided into seven separate boards.

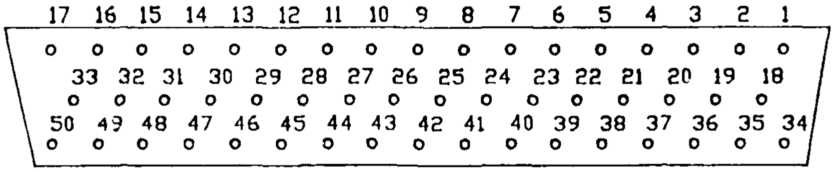
1. Main Computer: This board hosts the Motorola MC6801 microcomputer, the 8K x 8 byte RAM and transmitters and transceivers for the address and data buses.

2. Data Acquisition Board: This board holds the DAC0800 digital to analog converter, the two ADC0817 analog to digital converters and the latches required to buffer the input and output data lines from these devices.

3. Analog Interface: This board holds the transistors, buffers and differential amplifiers that are interfaced to the cells for the purpose of acquiring IV curves and steering annealing currents. There are four of these boards.

4. Anneal Interface: This board holds the transistors and multiplexers required to select the cell that will be annealed as well as to isolate the IV interface. Additionally, it holds the power conditioning components required to convert satellite power to the appropriate voltages needed to run the entire experiment. These components include the relays necessary to turn annealing and IV curve power on and off selectively. Annealing and IV curve acquisition cannot occur simultaneously.

Each of the boards measures 8x13 cm and is stacked one above the other. The hardware is placed in a machined aluminum box which measures 10x15x16 cm. A single 50 pin connector interfaces the box to the host main computer [Figure 6]. Thirty-two of the inputs are required for the connections from th 16 different solar cells. Three inputs are used for the power requirements of +28v, -28v and ground. One input is required for the interrupt line from the satellite computer which in turn is used to alert the Solar Experiment to serially download it's data. Thirteen



PINS 1,2 = CELL1	PINS 17,18 = CELL9	PIN 33 = +28V
PINS 3,4 = CELL2	PINS 19,20 = CELL10	PIN 34 = -28V
PINS 5,6 = CELL3	PINS 21,22 = CELL11	PIN 35 = SERIAL
PINS 7,8 = CELL4	PINS 23,24 = CELL12	PIN 36 = GND
PINS 9,10 = CELL5	PINS 25,26 = CELL13	PIN 37 = IRQ1
PINS 11,12 = CELL6	PINS 27,28 = CELL14	
PINS 13,14 = CELL7	PINS 29,30 = CELL15	REMAINING PINS
PINS 15,16 = CELL8	PINS 31,32 = CELL16	UNUSED

Figure 6 External Connector For NPS Solar Experiment

lines are not used. The last line is an output and is the serial port of the computer.

C. THESIS ORGANIZATION

This document is a detailed description of the design and construction of the NPS Solar Experiment. It's chapter divisions are primarily based on the components and activities of the several printed circuit boards comprising the experimental system.

Chapter II describes the design of the main computer .

Chapter III describes the data acquisition board and it's relationship to both the main computer and the analog interface.

Chapter IV discusses the anneal function and the different procedure used by the main computer to control the anneal circuitry.

Chapter V discusses the analog interface to the solar cell array.

Chapter VI describes the assembly language programs required to operate every function of the project.

Chapter VII provides operating instructions for starting up the experiment once it has been installed on it's host satellite.

Conclusions and recommendations are provided in Chapter VIII. These discuss the possible satellites that might host the NPS Solar Experiment and research that may be conducted by using this experiment.

II. MAIN COMPUTER

The main computer for the NPS Solar Experiment is completely contained on a single printed circuit board [Figure 7]. This board contains the following components:

1. Motorola MC6801 microcomputer.
2. Three 74HC373 data latches.
3. 74HC245 data transceiver.
4. CXK5864PN-15L 64K RAM.
5. 74HC138 address decoder.
6. 74HC04 inverter.
7. 4.915 Mhz quartz crystal oscillator.

A. MAIN COMPUTER DESIGN

The NPS Solar Experiement main computer is designed around the MC6801 8-bit microcomputer. This microcomputer is an enhancement of the MC6800 microprocessor which was used extensively in the Apple/Macintosh computer systems. The MC6801 is a natural choice for use due to its built in timer, built in serial ports and internal clock generator. Additionally, the MC6801 has 128 bytes of RAM and 2K bytes of ROM which is programmed by the factory to customer

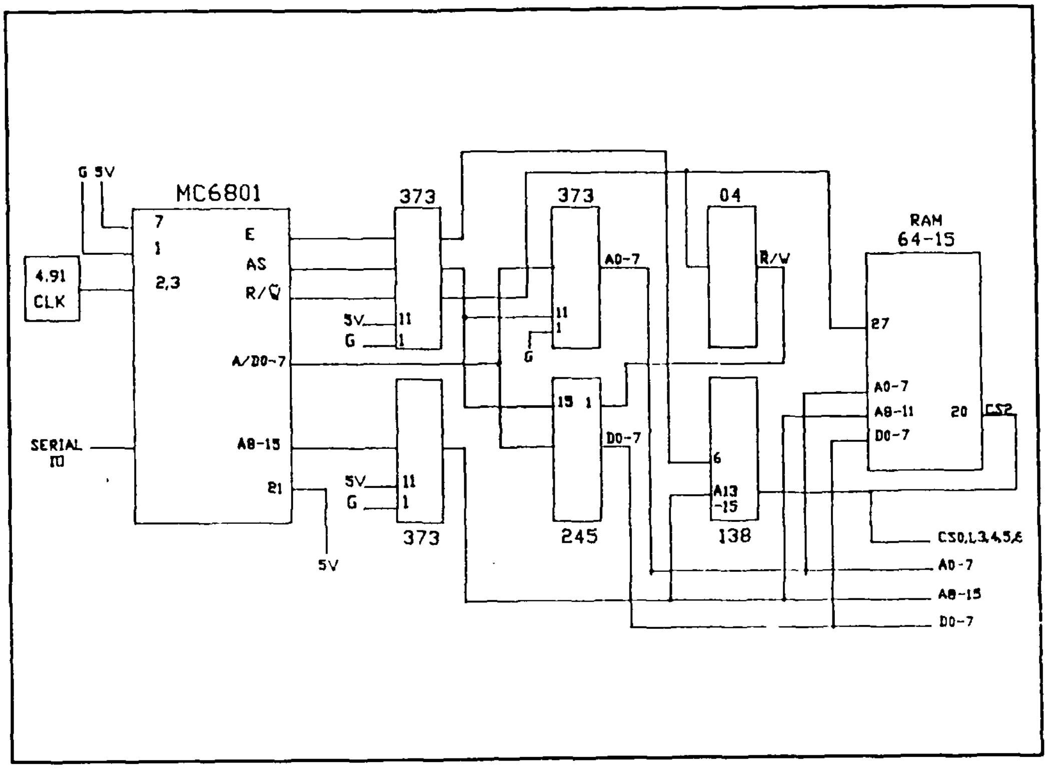


Figure 7   Satellite Main Computer

specifications. Each of these features is used by the Solar Experiment. (Extensive details of the MC6801 are provided in Appendix A.)

The MC6801 output leads have a fanout capability of one transistor-transistor logic (TTL) device. Because of this limitation, all output pins are fitted with buffers. Two 74HC373 are required to buffer address lines A8 to A15 and the address stobe (AS), the r ad/write (R/W) and the enable (E) pins. Additionally, the address lines A0 to A7 and the data lines D0 to D7 are multiplexed and must be de-multiplexed to separate them from each other. One 74HC373 is used for this purpose. The 74HC373 used here, adequately buffers the address lines but the data lines require an additional 74HC245 data transceiver. A transceiver is used in this case since the data lines are bidirectional whereas the address lines are unidirectional.

The 74HC138 address decoder provides the chip selects to the various components requiring control. These include the RAM, analog to digital converters, digital to analog latch and the anneal latch. It uses address lines A13, A14 and A15 to provide eight different 8K byte address spaces.

The 74HC138 control signal is always a low-logic level. This low-logic level will operate only on the chip select for the RAM. All other devices requiring chip selects respond to high-logic levels. The 74HC04 inverter is provided in the circuit for this purpose.

Two ten pin connectors are provided to allow for the data lines, address lines, clock signal and appropriate chip selects to be easily connected to the Data Acquisition Board. (The Data Acquisition Board will be discussed thoroughly in Chapter III.)

B. MAIN COMPUTER OPERATION

The purpose of the main computer is four-fold. First, the computer acts as a time-of-day clock. This function is required to ensure that the IV curve acquisition function and the anneal function occur on an appropriate schedule and that these functions operate for very discrete periods of time. This is an interrupt driven function, hence, it is independent of all other functions. This clock is driven by the highest priority interrupt to ensure that no other function causes the clock to lose time.

The second function of the computer is to incrementally step digital values on the data lines of the digital to analog converter. (This is the IV curve acquisition function.) It starts at $00 (hexadecimal 00) and steps one digit at a time up to $FF. This represents 256 distinct steps. The digital to analog converter (discussed in Chapter III) converts these digital signals to analog voltages. The analog voltages are applied to the base of the IV curve transistors which causes the solar cell current to incrementally increase. The voltages are measured by the analog to digital converters at the collector and emitter of the transistor for each step that the DAC makes. (The details of the operation of the transistor current control are covered in Chapter V.) These analog voltages are converted to digital (binary numbers) which are read by the main computer and then stored in RAM. These values represent different points along the IV curve.

The third function of the computer is to turn the anneal circuitry on and off. This is done by writing values to the anneal data latch on the Anneal Controller Board. This function is discussed in Chapter IV.

The last function of the computer is for data control. Upon receiving an appropriate computer command, the data that is stored in the RAM is transferred to the satellite's main computer for transmission to the ground. This is an interrupt driven function which is independent of the IV curve and anneal portions of the experiment.

B. MAIN COMPUTER DEVELOPMENT

The Motorola MC6801 has a 2K byte ROM built in the chip. When an over-the-counter chip is purchased, it comes with a monitor program prestored in the ROM. This ROM can also be programmed by the factory with a customers code which replaces the monitor. For the purposes of the NPS Solar Experiment, this monitor is not required.

To ensure that the code written for the MC6801 is operational prior to paying for a factory programmed version, Motorola makes a development microcomputer called the MC68701. The MC68701 microcomputer is an exact duplicate of the MC6801 except that the 2K ROM of the MC6801 is replaced by an erasable, programmable ROM (EPROM) and minor differences exist in the hardware required to program the EPROM. Essentially however, when the desired code is operational in the MC68701, it will be operational in the ROM of the MC6801. The use of the MC68701 as a developemnt tool is indispensable.

In order to program the MC68701, a programming computer had to be constructed. It is a modification of the satellite flight computer hence did not require excessive time expenditure in design and construction. Appendix B discusses the MC68701 and MC68701 EPROM programming computer in detail.

Additionaly, a high-reliability (Hi-rel) microcomputer suitable for a space mission had to be manufactured. Motorola Corporation manufactures the silicon chip portion of the processor with the customers ROM mask programmed and places the chip in a ceramic, milspec, side brazed package. This hermetically sealed chip is considered a space-qualified hardware item.

The main computer circuit board is shown in Appendix I.

III. DATA ACQUISITION

Data acquisition on the NPS solar experiment is handled by a single circuit board. This board contains the following components:

1. DAC0800 digital to analog converter.
2. Two ADC0817 16-channel analog to digital converters.
3. Two 74HC373 8-bit latches.
4. Two LF351 operational amplifiers.
5. LM334Z constant current source.
6. 7805 five volt regulator.
7. 74HC04 inverter.
8. DG506 analog multiplexer.

A. DIGITAL TO ANALOG CONVERSION

The purpose of the digital to analog system is to provide the stepping voltages to the bases of the IV curve transistors [Figure 8].

The DAC0800 D to A converter is designed to convert binary data between 0000 0000 and 1111 1111 to analog voltages. By selecting an appropriate voltage reference, accomplished with the LM334Z constant current source, the voltage can be tailored to match the beginning and ending voltages required for the IV curve. In this case, the value 0000 0000, when applied to the data pins of the D to A, produces exactly zero volts. When 1111 1111 is applied, the D to A produces

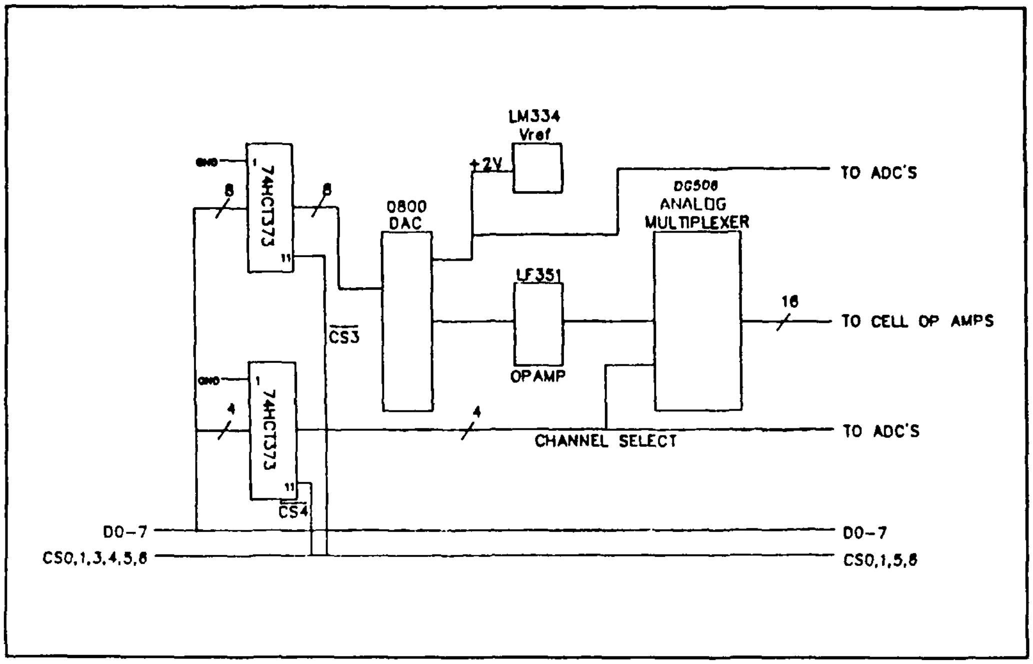


Figure 8 Solar Experiment Digital To Analog Circuitry

exactly 2.0 volts. Every binary digit in between is equivalent to 7.813 millivolts so precision stepping is possible.

When the microcomputer writes to address $8000 it is writing to a 74HC373 data latch. The data that is stored in this latch (between $00 and $FF) is the data that selects and holds the appropriate channel in both the DG506 multiplexer and the A to D converters. This channel select address must be latched to maintain uninterrupted output of the D to A to the appropriate IV transistor located on the analog interface.

When the microcomputer writes to address $6000 it is writing to a 74HC373 D to A data latch. This is required since the D to A converter cannot latch data.

It is a requirement that the 8-bit data from the microcomputer be held by the D to A converter until the A to D converters read the voltages from the IV transistors. Without the data latch, as soon as an attempt is made to write to the start pins on the A to D converters, the data to the D to A would be corrupted.

In overview, to send a single voltage value to the base of any transistor the following code would be required:

```
LDAA     #$07      ;Load internal register with binary 0111.

STAA     $8000     ;Store the 0111 in the channel select
                   ;latch. Channel eight is now open and will
                   ;remain open until a new value is sent
                   ;to the latch.

LDAA     #$EE      ;Load internal register with binary
                   ;1110 1110.

STAA     $6000     ;Store $EE in the D to A latch.
                   ;The D to A converter will now generate
                   ;1.859 volts and send it through channel
                   ;eight to the eighth solar cell in the array.
```

B. ANALOG TO DIGITAL CONVERSION

Two ADC0817 analog to digital converters are used in this design [Figure 9]. These converters have internal, 16-channel multiplexers eliminating the requirement for external devices. Each converter has 8-bit resolution and uses the same voltage reference as the D to A converter. These devices are accurate to plus or minus one least significant bit or 7.813 millivolts. The channel latch that selects the

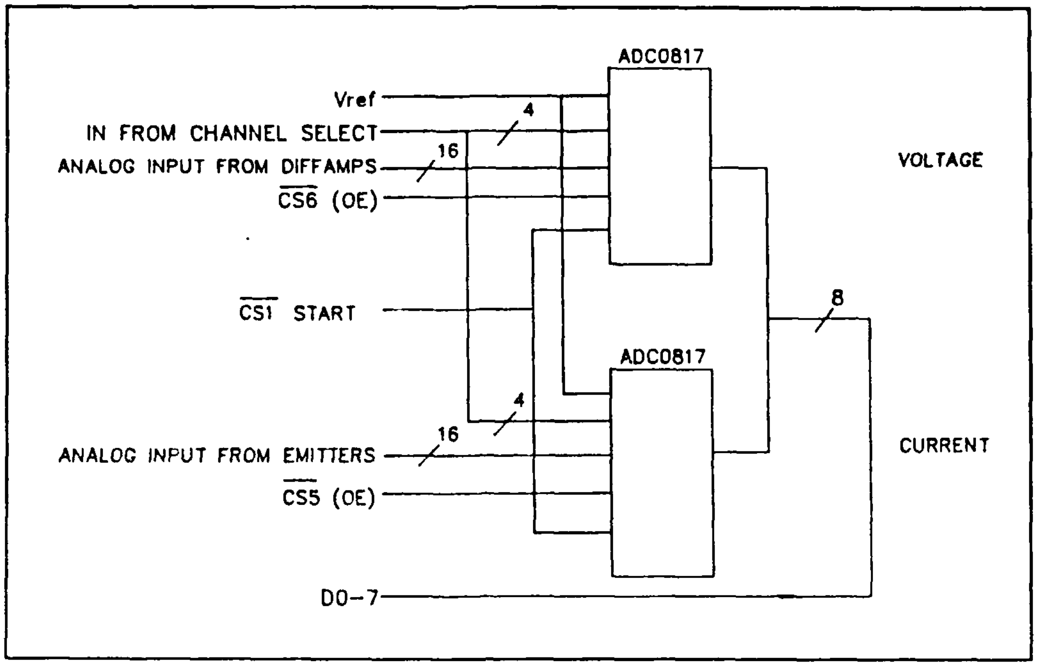


Figure 9 Solar Experiment Analog to Digital Circuitry

multiplexer channel for the DG506 is also connected to the channel selectors for the A to D converters. In the example above, when the eighth channel was selected for the DG506, the eighth channels were also selected in the A to D converters. In order to start the A to D converters, the start pins must be driven high. In this design, the start pins on both A to D's are tied together, hence, writing a one to a single start pin starts both A to D's. Address $2000 starts the A to D converters. Since the start pins are energized by an inverted chip select, it does not matter what data is written to it. (This is called a dummy write.) After waiting for 100 microseconds, the A to D converters will have finished converting the analog data to a digital pattern which can be read by the microcomputer. To read the voltage that the cell is producing, a read of address $A000 is required. To read the current from the cell, a read of address $C000 is needed.

Continuing with the same example as from above, the following code is required to retrieve the analog data:

```
WAIT      100        ;This is a macro that puts a 100
                     ;microsecond software delay in
                     ;the program.

LDAA      $A000      ;Load internal register with the data in
                     ;the first A to D converter.

STAA      RAM1       ;Store the data in a RAM location for
                     ;future use.

LDAA      $C000      ;Load internal register with the data in
                     ;the second A to D converter.

STAA      RAM2       ;Store the data in the next RAM location
                     ;for future use.
```

This process is repeated for each increment of the D to A converter and the entire process is then repeated for each cell in the array being tested. For a 16 cell test array the time required to complete testing of all cells is less than 0.5 seconds.

The main computer is simultaneously controlling IV curve data acquisition and annealing functions. The transistors for both of these functions are located on the analog interface board. Chapter IV will discuss the anneal controller prior to the discussion on the analog interface which is covered in Chapter V.

The data acquisition printed circuit board is shown in Appendix J.

IV. ANNEAL CONTROLLER

A single board contains the interface of the solar experiment main computer to the anneal transistors on the analog board [Figure 10]. This board contains the following components:

1. DG506 analog multiplexer.
2. 74HC373 data latch.
3. 74HC04 inverter.
4. LF351 operational amplifier.
5. 2N3405 transistor.
6. 12 volt, one ampere relay.
7. 12 volt, ten ampere relay.

The 74HC373 is used to hold the channel number for the DG506 analog multiplexer. Rather than using a variable reference as was done for the IV portion of the circuitry, the reference is a constant 12 volts. When a channel is selected, a 12 volt output is produced from that channel. Each channel is attached directly to the base of the anneal transistors which forces the corresponding transistor to become immediately active.

By addressing $1000, chip select zero goes low which is inverted by the 74HC04. This strobes the data latch and forces it to maintain the DG506's output enable either active or inactive and to activate a channel. This status is maintained until

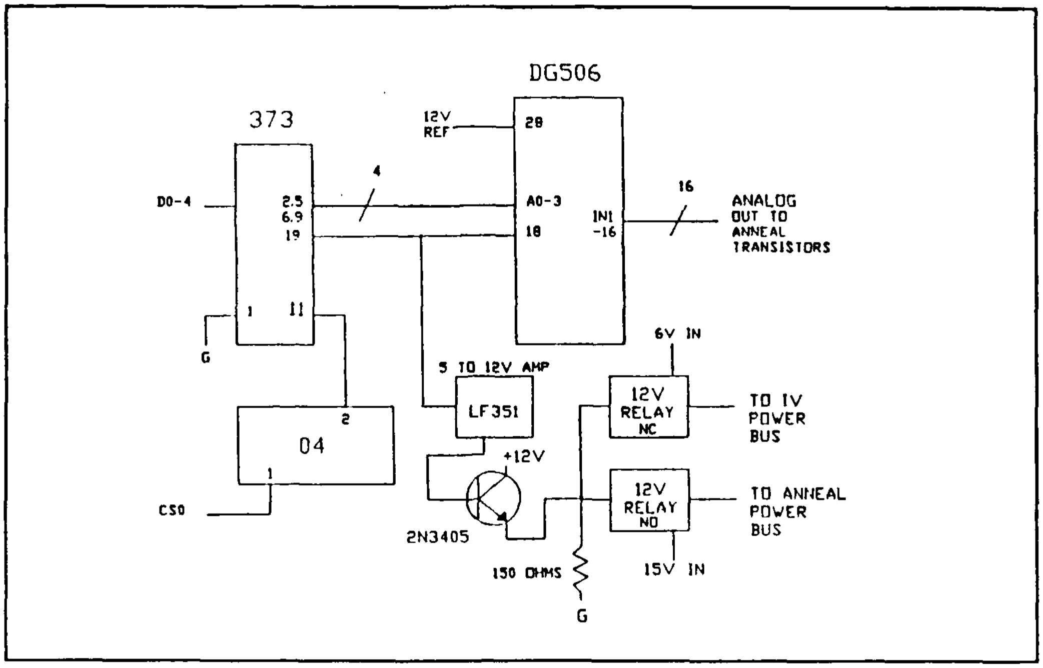


Figure 10 Solar Experiment Anneal Control Circuitry

new data is written to the latch. The most significant nibble (MSN) activates a single line connected to the enable output of the DG506. This line is also attached to an LF351 operational amplifier and tnen to the two 12 volt relays. Since only one relay at a time can be operational, an asserted MSN secures the six volt bus and activates the 15 volt bus. The least significant nibble of the byte (LSN) selects the channel. An example of the required program statements follow:

```
LDAA    #$14     ;Load internal register with binary
                 ;0001 0100.

STAA    $1000    ;Store the above value in the anneal
                 ;data latch. This causes 12 volts to
                 ;flow to the anneal transistor for cell
                 ;number five, switches the power bus from
                 ;six volts to 15 volts and activates the
                 ;output enable.
```

```
COMP     #24       ;This is a macro to compare the amount
                   ;of time that annealing on a single
                   ;cell has been active.  At 24 hours,
                   ;the annealing is secured.

LDAA     #$00      ;Load internal register with
                   ;binary 0000 0000.

STAA     $1000     ;Store the above value in the anneal
                   ;data latch.  The zero in the MSN of the
                   ;byte secures the output enable pin on the
                   ;DG506 as well as resets the relays to
                   ;the six volt bus.
```

The anneal control circuit board is shown in the Appendix L.

V. ANALOG INTERFACE

The heart of the entire experimental system is the analog interface circuitry shown in Figure 11. There are four boards that hold the IV acquisition and anneal transistor interfaces to the solar cell array. Each solar cell has its own discrete IV transistor, anneal transistor and differential amplifiers. These components are required to perform IV curve acquisition and long-term annealing. These functions cannot be multiplexed through an array. For acquisition of IV curves, the 15 volt source is secured. The presence of the diode in the 15 volt source line prevents the 15 volt power supply from being reversed by the six volt supply. With the 15 volt supply turned off and zero voltage applied to the base of the anneal transistor, the anneal circuitry is effectively isolated from the IV circuitry.

The six volt supply, after the voltage drop across the diode, reduces to 5.3 volts. With zero volts applied from the DAC multiplexer to the operational amplifier (buffer) and into the base of the IV curve transistor, the cell output is Voc (open-circuit voltage). Maximum voltage is produced when the current flowing in the cell is zero. The voltage measured at the collector is (5.3 + Voc). The LF351 operational amplifier is a differential amplifier which subtracts 5.3 from (5.3 + Voc) leaving only Voc to be measured by the voltage acquisition A to D converter. The current acquisition A to D converter measures the voltage across the transistor emitter resistance which is zero with zero volts applied to the base (transistor is

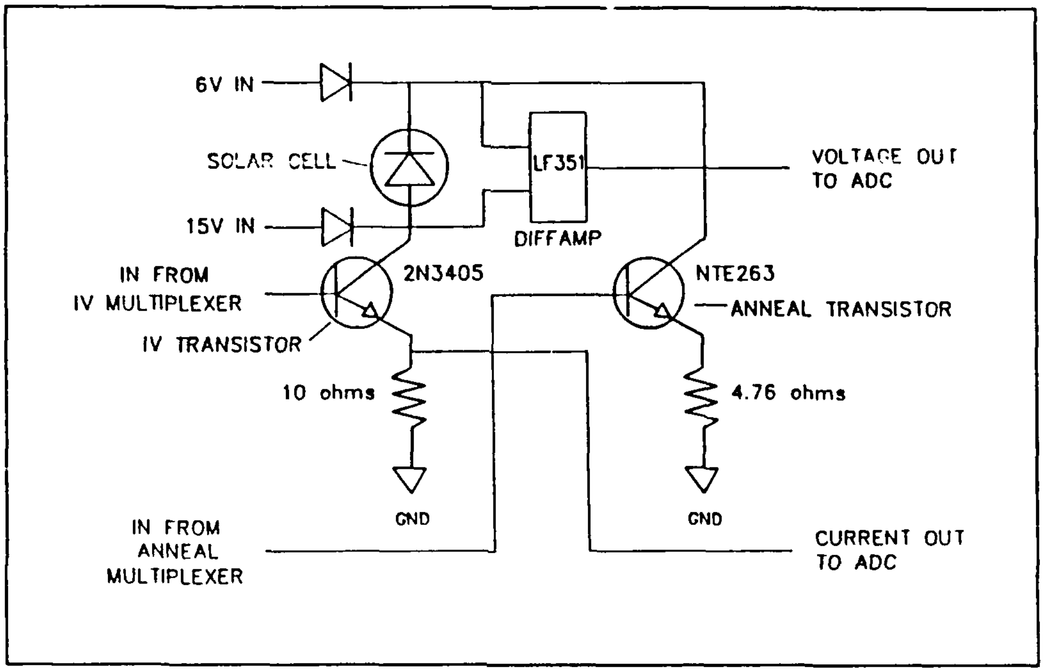


Figure 11 Solar Experiment Analog Interface

off). As the D to A converter steps the voltage (in 7.813 millivolt increments) into the base of the IV transistor, the transistor turns on and current begins to flow through the transistor emitter. Since the IV transistor has a Beta of 500, the current in the collector and the current in the emitter are equivalent to within 0.2 percent. By measuring the voltage in the emitter and dividing it by the resistor value (this is accomplished in software), the current produced by the solar cell is determined. Additionally, due to the high transistor Beta, the current in the base can be ignored.

As the current produced by the solar cell increases, the output voltage across the cell decreases. These characteristics can be observed in the cell IV curve. The differential amplifier will produce an output equal to 5.3 volts + cell voltage - 5.3

volts. When the voltage measured across the cell reaches zero, this corresponds to Isc (short circuit current). As the differential amplifier ouput becomes negative, a series of zeros will be read by the A to D converters since they are set up for basic unipolar operation.

By securing the six volt source and energizing the 15 volt source, the anneal segment of the analog interface becomes active. The only choices for the anneal transistor are on and off. When 12 volts is sent from the anneal multiplexer to the base of the transistor, the transistor becomes immediately active. The solar cell becomes forward biased. The cell current is controlled by the resistor in the emitter of the transistor. This current is 14.3 volts/4.76 ohms or three amperes. Since the anneal circuitry is digitally controlled ~nd on a 24 hour timer, no additional measurements are required during the anneal cycle.

VI. SOFTWARE CONTROL

The software controlling the NPS Solar Experiment is divided into three discrete segments. The first segment is the interrupt driven clock which maintains a series of counters in RAM. These counters include the number of days since the last IV curve acquisition cycle, the number of days since the last anneal cycle and an hour/minute/second counter. The hour/minute/second counter is a 24 hour clock that initializes to zero whenever the experiment is turned on. It's time does not correspond to any actual clock time.

The second segment is the main program which constantly polls the clock to determine whether the anneal process or IV curve acquiring function is scheduled. This polling is interrupted by the updating of the clock digits.

The third segment is the data dump interrupt driver. Upon the assertion of IRQ1, the main computer stops polling and serially transmits all the data it has stored in RAM. Upon completion of this function, polling recommences.

Additionally, these three software segments make extensive use of two libraries. The first library is the ‘macro’ library (MACLIB.ASM). This library contains short program units that are used for transmitting characters and strings of characters to a terminal, initializing various functions required by the microcomputer and simplifying the compilation of frequently repeated functions. (This library is shown in Appendix E.)

The second library is the 'symbol definition' library (SYMDEF.ASM) which contains the symbolic definitions for all memory storage locations, all hardware address locations and required terminal control characters. (This library can be located in Appendix F.)

These libraries are accessed by using the 'include' statement in the header of the program requiring library service.

Figure 12 shows the general logic of the Solar Experiment programs.

A. INTERRUPT DRIVEN CLOCK

The interrupt driven clock, SATCLK.ASM, uses the microcomputers on board timer and internal interrupt line IRQ2.

1. ON BOARD-TIMER

The MC6801 microcomputer contains an on-board timer which includes a 16-bit free-running counter, a compare register and a control/status register for programming its output [Figure 13]. The relevant registers are:

1. Counter Register: The counter register is the 16-bit free-running counter which resets to $0000. It is addressed as counter register most significant byte (CRM) at address $0009 and counter register least significant byte (CRL) at consecutive address $000A. It is a read-only register with one exception: any write to the counter register most significant byte will set the value of the 16-bit total to $FFF8. (This feature is used in the experiment clock.)

2. Output Compare Register: The output compare register is a 16-bit read/write register which resets to $FFFF. It is addressed as the output compare register most significant byte (OCRM) at address $000B and the output compare register least significant byte (OCRL) at address $000C.

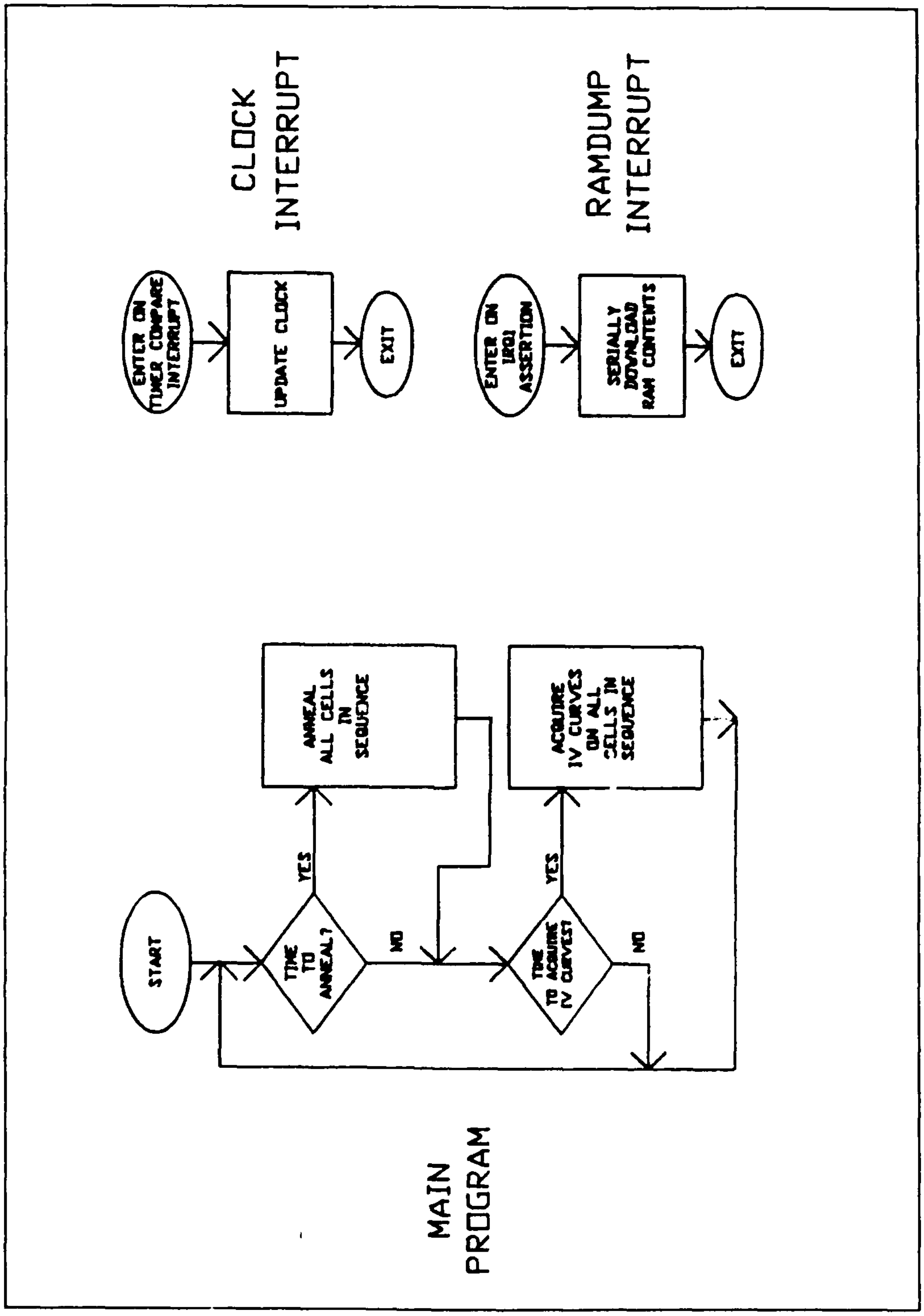

MAIN PROGRAM
START
TIME TO ANNEAL?
YES
NO
ANNEAL ALL CELLS IN SEQUENCE
TIME TO ACQUIRE IV CURVES?
YES
NO
ACQUIRE IV CURVES ON ALL CELLS IN SEQUENCE
CLOCK INTERRUPT
ENTER ON TIMER COMPARE INTERRUPT
UPDATE CLOCK
EXIT
RAMDUMP INTERRUPT
ENTER ON IRQ1 ASSERTION
SERIALLY DOWNLOAD RAM CONTENTS
EXIT

Figure 12 Satellite Software Flow Diagram

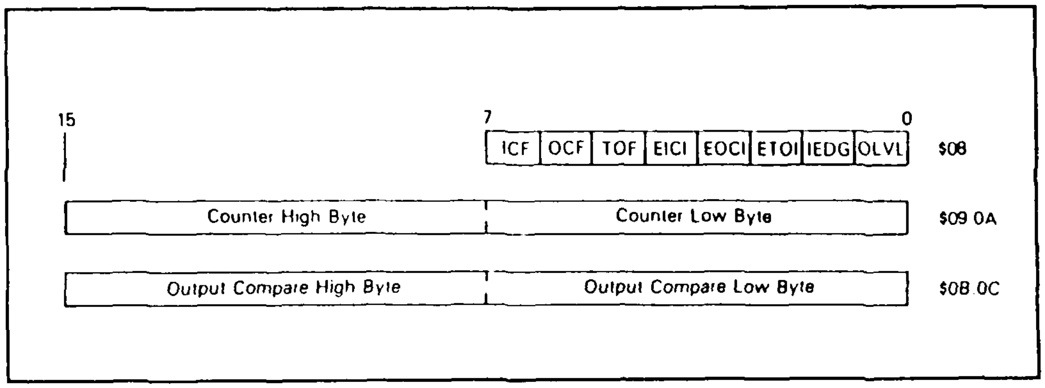


Figure 13 Timer Control Registers For MC6801 [adapted from Ref. 7]

3. Timer Control and Status Register: The timer control and status register(TCSR) contains the ccntrol bits for activating the timer. For this application the applicable bit is bit three. Setting this bit enables the output compare interrupt line IRQ2.

To activate the timer, $08 is written to the TCSR. With this write, any time that the free running counter (CRL+CRM) equals the output compare register (OCRL+OCRM), IRQ2 is forced low. Interrupt processing follows, which includes placing pertinent data on the stack and then fetching the address of the interrupt handler from address $FFF4 and $FFF5. The interrupt handler is the actual clock software for advancing digits in memory.

2. CLOCK AND INTERRUPT INITIALIZATION

The main program initializes the memory locations that will maintain the clock digits. Five memory locations are used to keep track of these digits. The labels and addresses of these locations are:

1. SECOND:	address $00B0;	second counter
2. MINUTE:	address $00B1;	minute counter

3. HOUR:	address $00B2;	hour counter
4. DAYANN:	address $00B3;	anneal day counter
5. DAYIV:	address $00B4;	IV day counter.

These memory locations are initialized to zero. A $08 is written to the TCSR to enable the output compare interrupt, and then the program simply runs in a loop waiting for the output compare interrupt to occur.

The interrupt handler proper, resides at address location $FB00. It is the program that advances the clock digits when necessary. This program starts by loading the output compare registers with $EFCE which is the value required to generate 50 milliseconds. The most significant byte of the counter register is dummy written which sets the counter to $FFF8. When the counter goes from $FFF8, through $0000 and finally reaches $EFCE, 50 milliseconds will have elapsed and an interrupt occurs. Since it takes 20 interrupts to make a single second, the interrupt handler contains a counter to ensure that a full second has elapsed prior to advancing a digit. If a full second has not elapsed, the handler returns to the main program. If the second has elapsed, the appropriate digits of the 24 hour clock are advanced. Additionally, two different day counters may be advanced. One day counter is for the number of days since the last IV curves were acquired and one day counter is for the elapsed time since annealing. For the flight experiment, no display is necessary and the clock is maintained in binary. Since the clock is interrupt driven by the highest priority interrupt, all processes done by the microcomputer (e.g., acquiring an IV curve) are interrupted every 50 milliseconds to advance the clock. Although the anneal timing cycle may be interrupted every 50 milliseconds, the anneal itself is not interrupted since the relays are latched and the latches are not disturbed during clock updates.

B. SATELLITE MAIN PROGRAM

The main program, MAIN.ASM, is responsible for initializing all required functions. The initializing steps follow:

1. Initialize the vectors for the clock and memory dump interrupt handlers.

2. Designate memory storage locations for acquired data and clock digits.

3. Initialize the clock memory to zero.

4. Align the serial port to transmit data at 9600 baud, no start bits, eight data bits and one stop bit.

5. Transmit $FF for 60 seconds to allow for it's initialization to be monitored to ensure operation.

6. Clear the interrupt bit and enable the output compare interrupt.

After initialization has occured the program commences polling the clock digits. It first checks to determine if the anneal day counter has reached day 180. If it has reached day 180, it commences sequentially annealling all 16 cells with the clock being monitored to ensure that each cell is annealed for exactly 24 hours. Upon completion of this cycle, it checks to see if it is time to acquire IV curves. If the anneal day counter is not at day 180, the program jumps to determine if the IV day counter has reached day ten. If it has, a series of IV curves are acquired and the data is placed in memory. That data will remain there until the host satellite's main computer calls for it or new IV curve data is acquired.

This program loops endlessly. If a spurious interrupt should occur in the system, vectors are provided in the main program to reinitialize the entire experiment.

C. INTERRUPT DRIVEN RAMDUMP PROGRAM

In response to IRQ1 being driven low by an external source, the vector for data transmission is fetched and the interrupt handler RAMDUMP.ASM takes control. This program is located in ROM at address $FD00. It goes to the first location in memory, $4000, gets the information at that location, and places it in the transmit register. This data is immediately transmitted on the serial output line. The program then increments the memory location and repeats the process until the end of memory is reached at address $4FFF. This represents 256 points per curve on 16 different cells. Upon completion of transmission, the RAMDUMP handler returns control to the main program.

The NPS Solar Experiment programs are shown in detail in Appendix D.

VII. SATELLITE EXPERIMENT OPERATING INSTRUCTIONS

The satellite experiment is generally autonomous with few connections to the satellites main computer. The required connections are referred to in Chapter I. After all these connections have been made, the power switch is turned on and the reset button is pressed. After the reset button has been pressed, the computer will transmit 60 seconds of serial data. This data can be monitored by placing a serial tester ground lead on the satellite power ground and the positive lead on the microplug labelled 'TEST'. This is the only indication that the experiment is operational until it commences transmission on-orbit.

The interface document for this experiment is not yet available. It is highly dependent on the satellite on which the experiment may fly. Depending on the interface design, one additional board may be required to be installed. This board (shown in Appendix M) is similar to the RS232 interface board shown in Figure 14. It contains the line drivers and receivers necessary to raise and lower the voltages of the serial data. In this design, the voltages are adjusted from five volts to 12 volts on the line driver and from 12 volts to five volts on the line receiver. This board was used extensively during the testing of the flight computer to interface it with an IBM PC. Since only the serial line out of the flight experiment is used,

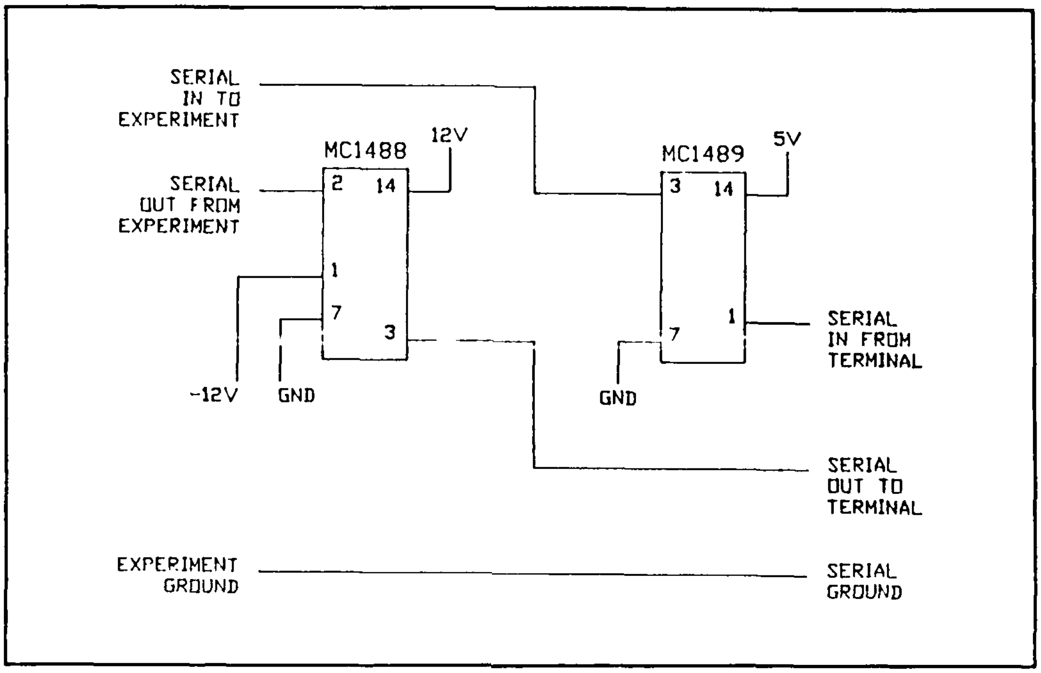


Figure 14 RS232 Interface

only the line driver chip is needed. During the testing phase of the experiment, both driver and receiver chips had to be in place.

There is a possibility that the host satellite will require an RS422 interface. If this is the case, a simple chip change to a Motorola MC3487 vice MC1488 will correct the discrepancy.

VIII. CONCLUSIONS AND RECOMMENDATIONS

A. POTENTIAL FOR USAGE

As was stated in the introduction, any extension of the life of a solar cell/array will also extend the life of the satellite bus. Although this experiment was designed to test and anneal cells one at a time, it can be easily adapted to testing and annealing sections of arrays, thereby minimizing the amount of annealing circuits necessary to do the job.

The capability of recording the performance and degradation of cells on-orbit is possible with this equipment. This function alone will be of some interest to space environment physicists and certainly to manufacturers of satellites and satellite solar arrays.

B. ANTICIPATED FLIGHT EXPERIMENTS

Two satellites are currently being considered for flight with the NPS Solar Experiment.

1. PANSAT

Pansat is a Naval Postgraduate School experimental satellite. It is a 'Get Away Special' project hence, it will be flying in low-earth orbit. This orbit is not expected to cause degradation in the solar cells. Thus, the anneal portion of the experiment will not be utilized. The experiment will be limited to acquiring IV curves only.

2. Applied Physics Laboratory Satellite

The Johns Hopkins University Applied Physics Laboratory is currently developing a satellite for orbiting through the Van Allen belts. Designers of the satellite expressed interest in including this experiment on-board to test different state-of-the-art solar cells. Due to expected degradation of the cells in this orbit, full use of the experiment IV measurement and current annealing would provide valuable information on radiation effects and prove the new current annealing theory.

C. FUTURE RESEARCH

1. It has been shown in the laboratory that current annealing of GaAs cells at low temperature restores much of the power that was lost due to radiation damage of the type that is encountered in the space environment. Topics that bear further examination are:

a. If a GaAs cell is not heavily irradiated (e.g., to a fluence of 1 x 10^{14} instead of 1 x 10^{15} electrons/cm^2) will annealing the cell restore a higher percentage of the lost power?

b. Is there a practical upper limit to the number of times that a cell can be current annealed?

c. How would the newest generation of Indium Phosphide (InP) solar cells behave when this process is used?

2. During certain portions of a satellite's flight (particularly low-earth orbits) portions of arrays are eclipsed by the earth. In such cases, when part of the array is actively providing power to the satellite, the eclipsed or shadowed cells experience reverse-biasing effects. Cell characteristics during these power transitions, which can be measured using this experiment, are of some interest.

D. RECOMMENDATIONS

1. The resolution of the analog to digital and digital to analog conversion segments in the experiment are eight bits. These devices should be replaced by 12-bit devices. Resolution could then be achieved in the 0.5 to one millivolt range vice the seven to eight millivolt range.

2. Actively search for additional buses on which this experiment can fly. Testing of cells in a wide variety of orbits will give much additional information on the effects of the space environment on solar cells and arrays.

APPENDIX A.

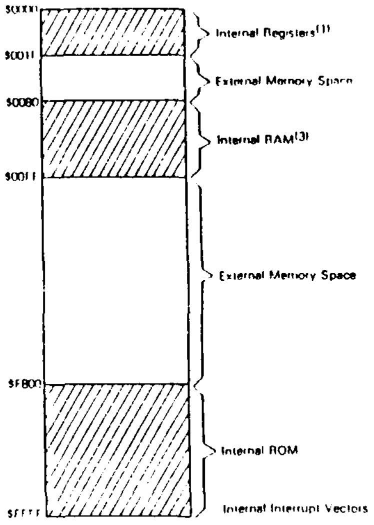
I. MOTOROLA MC6801 MICROCOMPUTER

The MC6801 is a 40 pin microcomputer capable of operating in eight distinct modes. Each mode takes advantage of a different feature of the microcomputer and uses a different memory mapping. The selection of the operating mode depends on operator requirements [Figure 15]. In the case of the NPS Solar Experiment, mode six is used for several reasons:

1. The internal ROM contains the program for the project rather than having external ROM.
2. The internal RAM is used to hold the stack and static memory locations.
3. External memory space is required to maintain data from the experiment.
4. Internal interrupt vectors are required since no external ROM is available.

The pin arrangement of the MC6801 varies also depending on the mode of operation [Figure 16]. In the figure, the first diagram shows the pinout in terms of ports. P10-17, P20-24 and P30-47 are all input/output pins and are defined both by choosing the mode and by enabling registers with software. The second diagram shows the pinout in mode six. In this diagram, a more traditional presentation of address and data buses is displayed.

The address lines A0-A7 are the same lines as the data lines D0-D7. It is necessary to de-multiplex the address and data buses. The address strobe (AS) pin



Notes

(1) Excludes the following addresses which may be used externally $04, $06, $0F

(2) Address lines A8-A15 will not contain addresses until the Data Direction Register for Port 4 has been written with "1"s in the appropriate bits. These address lines will assert "1"s until made outputs by writing the Data Direction Register

(3) Assumes RAME (RAM Enable bit) is set

Figure 15 Addresses for Operating Mode Six
[Ref. 7: p. 2-16]

is provided for this function. In the NPS Solar Experiment, a 74HC373 latch is used for this purpose [Figure 17]. The address/data bus is connected to the inputs of the 74HC373 and due to the timing of the address strobe, the address alone is passed to the outputs [Figure 18]. Note in the figure that during the negative level of the E (clock) pin, AS is high. Address becomes valid and remains valid during the falling edge of AS which holds the address in the latch until the next occurrence of AS. During the time between address latches, data becomes valid during the falling edge of E thereby completing the de-multiplex process.

Pins 39 and 38 labeled SC1 and SC2 in Figure 16, are configured as AS and

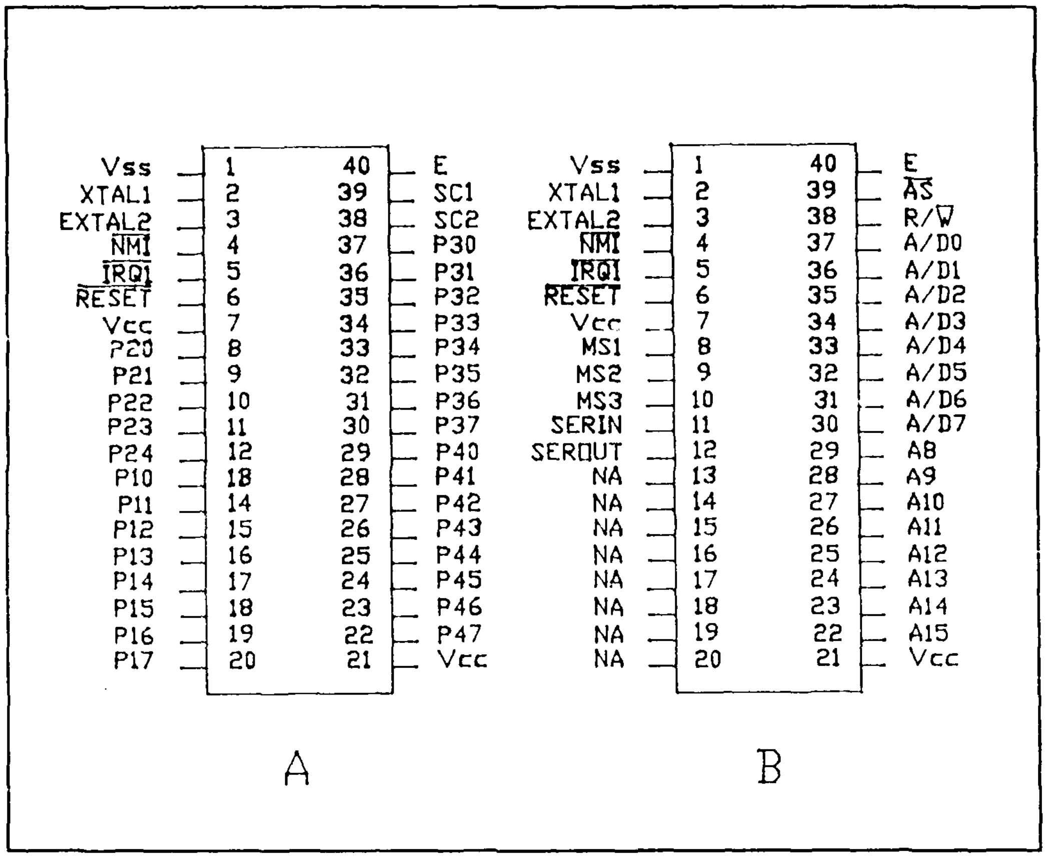


Figure 16 MC6801 Pin Arrangement [Ref. 7: p. 3-1]

A. General pin arrangement
B. Mode six pin arrangement

R/W in mode six. Pins 11 and 12 are serial in and out respectively. (Interface to RS232 or RS422 protocol requires additional line drivers and receivers.) Pins 13 through 20 are unused in the NPS design.

Pins eight through ten are the pins used to program the mode of the processor. (Figure 19 shows Motorola's recommend circuit.) The reset time constant and mode programming functions are highly associated with each other. The reset pin (pin six) must be held low for a sufficient period to allow the quartz oscillator to

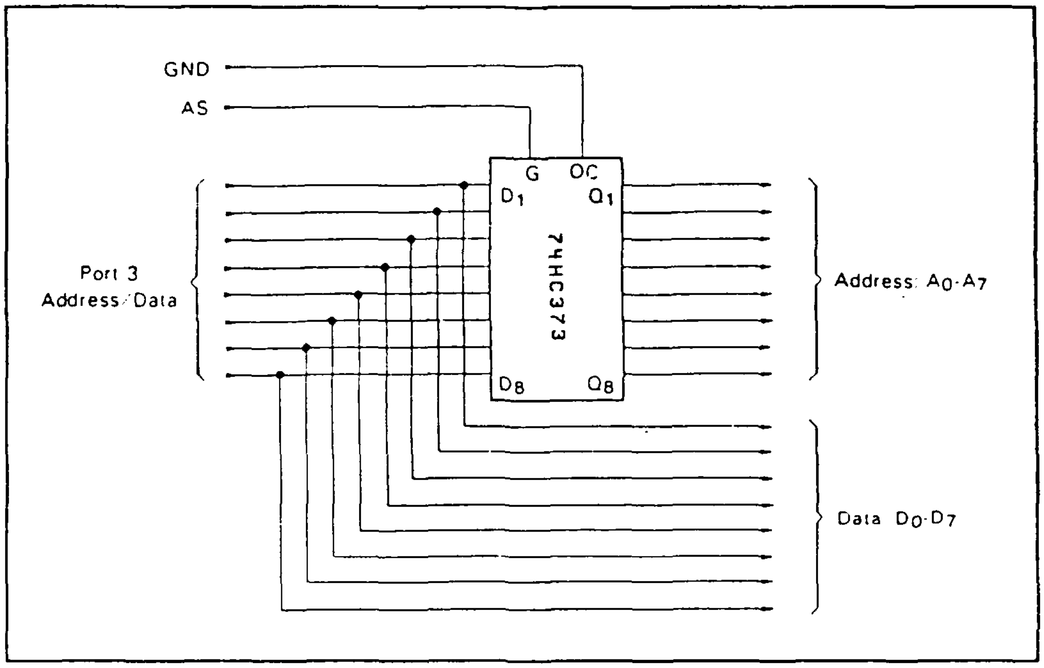


Figure 17 De-multiplexing Address/Data Lines
[Ref. 7: p. 3-43]

stabilize. (Normally ten clock cycles will be sufficient.) When the reset line goes high, the operating mode is latched and essentially cannot be changed unless another reset occurs. The significance here is that pins eight through ten are also used for port two functions and cannot simply be tied high and low if these functions are to be used. Although port two is not utilized in the NPS Solar Experiment, normal reset circuitry is provided to prevent spurious mode selection during reset. For mode six selection, pin ten and pin nine are kept high while pin eight is connected to low.

The MC6801 has 21 internal registers which are used to control various functions within the processor. For the NPS Solar Experiment, the following

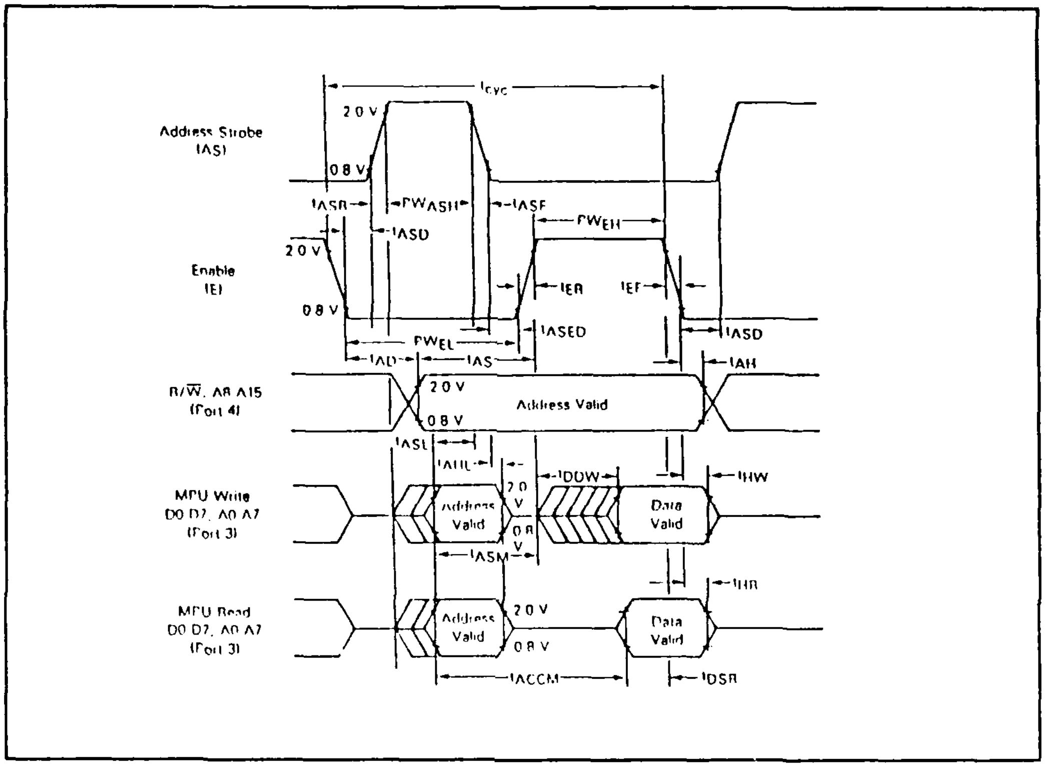


Figure 18 Timing Diagram for Multiplexed Mode Six [Ref. 7: p. 3-42]

registers are applicable:

1. Port Four Data Direction: This register configures port four as an output port and assigns the port as address lines eight to 15.

2. Timer Control And Status: This register enables timer interrupts and contains the flags that indicate a certain function has occurred within the timer.

3. Counter Register Most Significant Byte: This register contains the most significant byte of the 16-bit free running counter.

4. Counter Register Least Significant Bytc. This register contains the least significant byte of the free running counter.

5. Output Compare Register Most Significant Byte: This register contains the most significant byte of the number to be compared to the counter register.

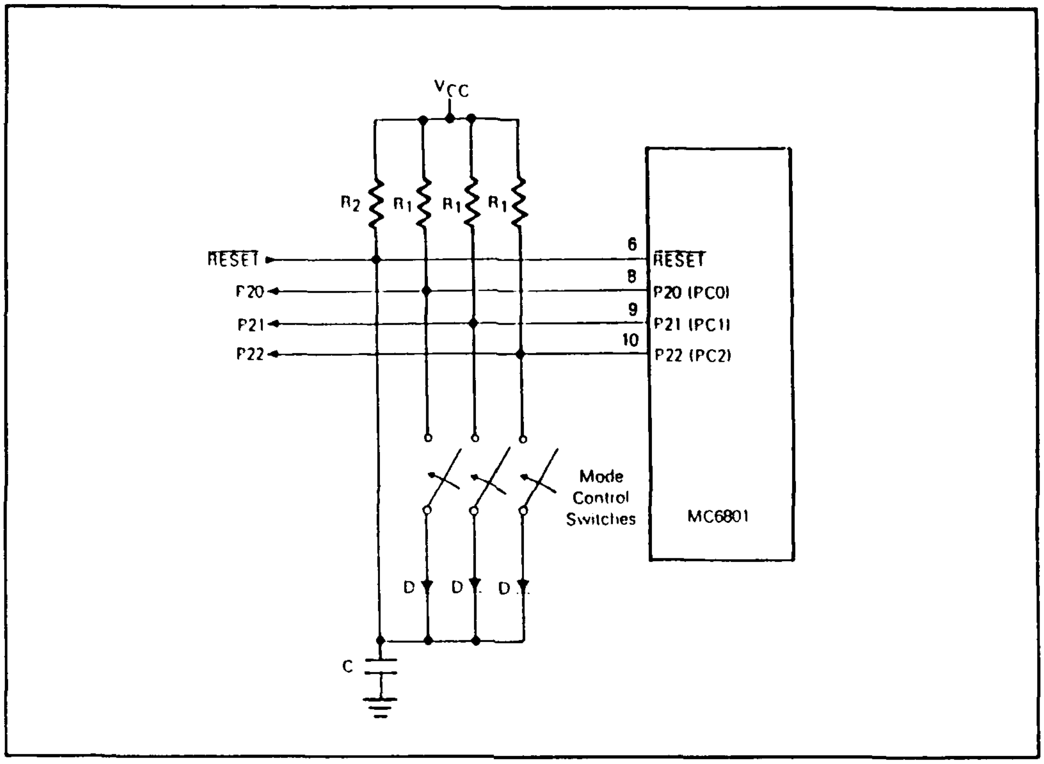


Figure 19 Mode Selection Circuitry [Ref. 7: p. 3-10]

6. Output Compare Register Least Significant Byte: This register contains the least significant byte of the number to be compared to the counter register.

7. Rate And Mode Control: This register sets the serial port baud rate and format of transmission.

8. Transmit/Receive Control And Status: Information in this register determines the existence of data in the receive register and whether the transmit register is empty or not.

9. Serial Receive Data Register: This register contains data coming in serially from external sources.

10. Transmit Data Register: This register holds data to be transmitted serially.

11. RAM Control Register: This register allows for internal RAM to be enabled or disabled from the address map.

The minimal design, represented by the Solar Experiment main computer, is possible due to the use of MC6801 assembly language. By using assembly instead of a high-level language, all required functions are handled with a ten-fold reduction in ROM size requirement.

The MC6801 uses an enhanced MC6800 instruction set which takes advantage of the MC6801's greater throughput. (See Reference 7 for details of the enhanced assembly language.)

Appendix I contains the printed circuit board which is suitable for immediate manufacturing of additional units.

APPENDIX B.

I. MOTOROLA MC68701 PROGRAMMABLE MICROCOMPUTER

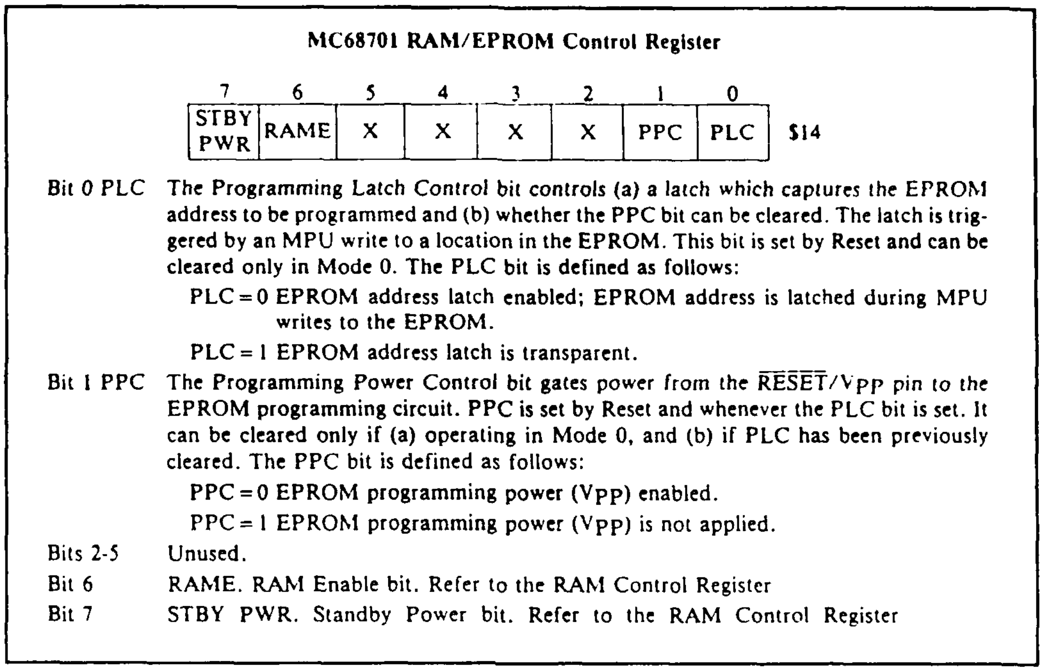
A. MC68701 VERSUS THE MC6801

The MC68701 is functionally similar to the MC6801 with the major difference being that its ROM is replaced by an EPROM in the exact address locations. Externally, the reset pin (pin six) is also used as the programming voltage input pin to the EPROM. Internally, the RAM control register contains additional bits for directing programming voltage (+21 volts) to the internal EPROM [Figure 20].

The MC68701 is an indispensable tool for developing the hardware and software necessary to generate a working project. Once the programs have been stored in the EPROM and operate correctly, the user can be confident that those programs will operate in the ROM of a MC6801.

B. PROGRAMMING THE MC68701

The MC68701 cannot be passively programmed in the manner that a standard EPROM can (e.g., a 2764). The MC68701 takes an active role in its own programming. This is necessary since the small number of pins of the microcomputer are used for multiple functions. The programming voltage must be steered by the microcomputer from the reset pin. The steering of this voltage is controlled by the RAM/EPROM control register located at address $14. Bit one is the applicable bit in the register. It is called the Programming Power Control bit



MC68701 RAM/EPROM Control Register

7	6	5	4	3	2	1	0	
STBY PWR	RAME	X	X	X	X	PPC	PLC	$14

Bit 0 PLC The Programming Latch Control bit controls (a) a latch which captures the EPROM address to be programmed and (b) whether the PPC bit can be cleared. The latch is triggered by an MPU write to a location in the EPROM. This bit is set by Reset and can be cleared only in Mode 0. The PLC bit is defined as follows:

PLC = 0 EPROM address latch enabled; EPROM address is latched during MPU writes to the EPROM.

PLC = 1 EPROM address latch is transparent.

Bit 1 PPC The Programming Power Control bit gates power from the $\overline{\text{RESET}}$/Vpp pin to the EPROM programming circuit. PPC is set by Reset and whenever the PLC bit is set. It can be cleared only if (a) operating in Mode 0, and (b) if PLC has been previously cleared. The PPC bit is defined as follows:

PPC = 0 EPROM programming power (Vpp) enabled.

PPC = 1 EPROM programming power (Vpp) is not applied.

Bits 2-5 Unused.

Bit 6 RAME. RAM Enable bit. Refer to the RAM Control Register

Bit 7 STBY PWR. Standby Power bit. Refer to the RAM Control Register

Figure 20 MC68701 RAM/EPROM Control Register [Ref. 7: p. E-8]

(PPC). When a ‘0’ is applied to this bit the power is enabled. When a ‘1’ is applied it is disabled. When the microcomputer resets, bit one initializes to a ‘1’.

The Programming Latch Control bit (PLC) is also an active participant. This bit is bit zero and initializes to a ‘1’. The PLC bit controls a latch that captures the address of the byte to be programmed within the EPROM. (The only time this bit can be cleared is when the microcomputer is in mode zero.)

The MC68701 has the same modes of operation as the MC6801. For programming, the MC68701 microcomputer must be reset and initialized in mode zero. At this point the MC68701 fetches a vector for its first instruction from location $BFFE and $BFFF. This address must be present in an external ROM and must point to the first address of the instruction set that needs to be run to program the EPROM. (Note here that depending on how the address space is designed, there may be several different addresses that can apply to the $BFFE and $BFFF location.) In this implementation, with a 74HC138 for the decoder and address pins A13, A14 and A15 used for selection, chip select five is driven by any address from $B000 to $BFFF.

In the design discussed here, the code that is to be placed into the microcomputer EPROM is programmed into a standard 2732 EPROM with any commercial EPROM programmer. (The only significance to using the 2732 is that they are readily available in the laboratory.) Only the last 2K bytes of the EPROM are used since this corresponds to the size of the onboard EPROM. (There is no requirement to have the program to be copied into the MC68701 in a separate EPROM. Certainly the program that runs the processor and the program to be copied can be on the same EPROM but the additional difficulty of having to reprogram the monitor program each time makes the small amount of additional hardware worthwhile.)

Reference 7 provides instructions for programming the internal EPROM. The following are the applicable steps:

1. The microcomputer is initialized to mode zero.

2. A '0' is written to PLC and a '1' to PPC at address $14. (The writing of a '1' initially to PPC is precautionary since it should initialize to a '1'.)

3. A byte is read from the EPROM to be copied and placed in accumulator A in the MC68701.

4. This byte is written to the next on-board location to be programmed.

5. A '0' is then written to the PPC and a 50 millisecond time delay is generated by the on-board timer to allow the EPROM time to program .

7. Return to step two for each byte to be programmed. (Time is saved by having a check loop in the software to avoid programming '0"s.) Note also that most EPROMs erase to $FF instead of zero. $FF is a valid instruction in the MC6801/701 instruction set so care must be taken to ensure that $FF is not left floating unaccounted for.

8. '1"s are written to PPC and PLC.

Once the microcomputer has been programmed, it can be removed and tested in appropriate circuitry.

The printed circuit board is displayed in Appendix H. It is suitable for direct use in manufacturing additional boards.

C. MC68701 PROGRAMMING SOFTWARE

PROGRAM.ASM is the resident program in the MC68701 programming board that copies the information from a 2732 EPROM in ROM slot two, to the internal EPROM on board the microcomputer. The microcomputer must be initialized in mode zero. Initially the program does a blank check where every ROM location is read and then downloaded to the screen. Next the programming loop programs every internal EPROM location except when a zero occurs. The program may be run a second time in the blank check phase to ensure that programming has occurred. Since the blank check is a hex dump, the program in the EPROM can be examined for accuracy.

```
••••••••••••••••••••••••••••••••••••••••••••••••••••••••••

    INCLUDE MACLIB.ASM        ;ENABLE PROGRAM TO READ MACRO
                              ;LIBRARY.
    INCLUDE SYMDEF.ASM        ;AND SYMBOLIC DEFINITION
                              ;LIBRARY.
        DATA

••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
PREPARE ADDRESSES FOR APPROPRIATE MESSAGES TO USER.
••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
        ORG         $BE00     ;PLACE DATA IN ROM AT THIS
                              ;LOCATION.

MSG1        BLKB      25,$0A
            BYTE      CR,'                   SE R'
            BYTE      'ESEARCH',CR,LF
            BYTE      '                 68701'
            BYTE      'INTERNAL EP'
            BYTE      'ROM PROGRAMMER',CR,LF,LF
            BYTE      '                  LT.  GARY'
            BYTE      'SOMMERS, MARCH 1989'
            BYTE      CR,LF,LF,LF,LF,LF,LF,LF,LF,LF,NUL
MSG2        BYTE      'PRESS > TO COMMENCE BLANK'
            BYTE      'CHECK'
            BYTE      CR,LF,NUL

MSG3        BYTE      CR,LF,LF,'SET SW3 OFF, SW1 ON'
            BYTE      'AND',CR,LF,NUL

MSG4        BYTE      'PRESS > TO COMMENCE'
            BYTE      'PROGRAMMING'
            BYTE      CR,LF,LF,LF,LF,LF,LF,NUL

MSG5        BLKB      17,$0A
            BYTE      CR,LF,LF,LF,LF,'SET SW1 OFF '
            BYTE      'THEN SW3 ON',CR,LF,LF,NUL

MSG6        BYTE      'PROGRAMMING COMPLETE',CR,LF
            BYTE      BEL,BEL,BEL,BEL,BEL,BEL,BEL,NUL

WAITM       BYTE      CR,LF,'PRESS > TO'
            BYTE      ' CONTINUE',CR,LF,LF,NUL
```

```
ENDMS       BYTE            CR,LF,LF,'BLANK CHECK IS'
            BYTE            'COMPLETE',CR,LF,LF,LF,NUL

       DATA

..........................................................
ADDITIONALLY, PUT PROGRAM STARTING ADDRESS IN THE FIRST
ADDRESS THAT THE PROCESSOR LOOKS AT UPON RESET.
..........................................................

       ORG  $BFFE

            WORD   $B000

       CODE

..........................................................
COMMENCE PROGRAM AT FIRST ADDRESS SPECIFIED.
..........................................................

       ORG  $B000

       INIT                    ;MACRO TO INITIALIZE SERIAL
                               ;PORT TO 9600,N,8,1, SET
                               ;PORT FOUR TO OUTPUT ADDRESSES
                               ;AND ENABLE INTERNAL RAM.
                               ;Vcc STANDBY MUST BE HIGH.

       LDS       #STACK        ;STACK IS IN INTERNAL RAM.

       LDAA      #$00          ;ENSURE COUNTER FOR PROGRAM
                               ;LOCATION STARTS AT ZERO.

       STAA      SRA           ;INDICATOR TO THE SCREEN STARTS
                               ;AT ZERO.

       STAA      SRB           ;SRA AND B ARE STORAGE
                               ;LOCATIONS FOR THE ON SCREEN
                               ;COUNTER.

       TXSTR     MSG1          ;USE MACRO TO SEND STRINGS TO
                               ;SCREEN.
```

```
        TXSTR     MSG2

        ECHO      '>'        ;> CHARACTER STARTS PROGRAM.


••••••••••••••••••••••••••••••••••••••••••••••••••••••••
THE FOLLOWING MACRO CALL DOES A HEXADECIMAL MEMORY
DUMP FROM START ADDRESS TO END ADDRESS INDICATED BY THE
ARGUMENTS.  THIS IS THE BLANK CHECK.  EPROM ERASES EACH
LOCATION TO 00.
••••••••••••••••••••••••••••••••••••••••••••••••••••••••

        TXCHAR    #CR
        TXCHAR    #LF
        RAMDMP1   EPRST,EPREND
        TXSTR     ENDMS

••••••••••••••••••••••••••••••••••••••••••••••••••••••••
COMMENCE PROGRAMMING OF INTERNAL EPROM.  THIS PORTION OF
THE PROGRAM IS FUNDAMENTALLY PROVIDED BY MOTOROLA IN
THEIR MANUAL 'MC6801 REFERENCE MANUAL', MC6801RM(AD2), PG'S
E12 TO E14.  THE ONBOARD TIMER IS USED TO PROVIDE A 50 ms
DELAY TO ALLOW TIME FOR EACH EPROM LOCATION TO PROGRAM.
••••••••••••••••••••••••••••••••••••••••••••••••••••••••

        TXSTR     MSG3       ;TRANSMIT ADDITIONAL MESSAGES.
        TXSTR     MSG4

        ECHO      '>'        ;> CHARACTER STARTS PROGRAM.

        TXCHAR    #CR
        TXCHAR    #LF

        LDX       #ROMST     ;PUT COPY ROM START LOCATION IN
                             ;INDEX REGISTER

        STX       SAVER      ;SAVE THIS LOCATION IN MEMORY.

        LDX       #EPRST     ;DO THE SAME FOR THE FIRST
                             ;EPROM LOCATION.
        STX       SAVEEP
```

```
*********************************************************
THE FOLLOWING SIX LINES DISPLAY THE LOCATION IN THE EPROM
THAT IS CURRENTLY BEING PROGRAMMED. 0000 IS THE FIRST
LOCATION WHILE $07FF IS THE LAST.
*********************************************************

LOOP1:  TXCHAR   #CR       ;MOVE DOWN ONE LINE TO
                           ;PREVENT OVERWRITE.

        RAMDMP   SRB,SRA   ;HEXDUMP THESE MEMORY
                           ;LOCATIONS
                           ;RAMDMP IS A MACRO.

        INC      SRA       ;INCREMENT THE BYTE AND THEN
                           ;CHECK FOR OVERFLOW TO THE
        BEQ      INCRB     ;NEXT BYTE TO KEEP COUNTER
                           ;ACCURATE.

        BRA      CONTIN    ;FINAL NUMBER IS $07FF.

INCRB:  INC      SRB       ;INCREMENT UPPER DIGITS IF
                           ;APPROPRIATE.
CONTIN: LDAA     #$FE      ;REMOVE Vpp AND SET THE ADDRESS
                           ;LATCH.
        STAA     RAMCR

        LDX      SAVER     ;GET A ROM LOCATION.

        LDAA     X         ;PUT ITS CONTENTS IN ACC A.

        INX                ;MOVE TO NEXT ROM LOCATION FOR
                           ;NEXT TIME.
        STX      SAVER     ;STORE THE POINTER.

        LDX      SAVEEP    ;NOW GET THE EPROM LOCATION TO
                           ;PUT IT IN.

        STAA     X         ;STORE ACC A IN THAT EPROM
                           ;LOCATION.

        CMPA     #$00      ;IF ITS A ZERO
        BEQ      NEXT      ;DON'T PROGRAM IT.
```

```
         LDAA     #$FC      ;ENABLE Vpp (21 TO 22 VOLTS ON
                            ;RESET PIN).
         STAA     RAMCR

         LDD      #WAIT     ;PUT AMOUNT OF DELAY INTO
                            ;DOUBLE ACCUMULATOR.
         ADDD     CRM       ;ADD TO THAT THE CURRENT STATUS
                            ;OF THE TIMER.

         CLR      TCSR      ;CLEAR ALL FLAGS IN TCSR.

         STD      OCRM      ;PUT DOUBLE ACC IN OUTPUT
                            ;COMPARE REGISTER.

         LDAA     #$40      ;PREPARE FOR FLAG TEST.

TEST:    BITA     TCSR      ;HAS NUMBER OF WAIT CYCLES
                            ;PAST?
         BEQ      TEST      ;IF NOT, CHECK AGAIN.

NEXT:    CPX      #EPREND   ;IF INDEX IS AT LAST LOCATION
                            ;GO TO TERMINAL MESSAGES,
         BEQ      JUMP      ;OTHERWISE, RETURN FOR
         INX                ;FOR ADDITIONAL LOCATIONS.

         STX      SAVEEP

         JMP      LOOP1

JUMP:    LDAA     #$FF      ;REMOVE Vpp AND SECURE LATCH.
         STAA     RAMCR

TERM:    TXSTR    MSG5      ;SEND AN END INDICATION TO
                            ;THE SCREEN.
         TXSTR    MSG6      ;SEND SECOND END MESSAGE
                            ;TO THE SCREEN.

FINISH:  NOP                ;COMMENCE TERMINAL LOOP.
         BRA      FINISH

         END
```

D. OPERATING INSTRUCTIONS FOR THE MC68701 EPROM PROGRAMMER

As discussed in earlier chapters, the MC68701 has an EPROM integrated into the microcomputer. This EPROM has a start address of $F800 and an end address as $FFFF. The programmer board has a ROM location at $4000 to $4FFF. This ROM location is designed to be used by a 2732 EPROM. The program that is to be inserted into the MC68701 is first placed into the 2732 by a commercial EPROM programmer, with the first address programmed at location $0800. The flight computer will recognize this address as $4800. The last 16 bytes in the 2732 must be programmed with the interrupt vector table. At a minimum, the final two bytes must be programmed with the value $F800, which is the vector for the EPROM start address in the MC68701. The rest of the vectors are inserted into the vector table if they are to be used. It is a good practice to supply values in these unused vector slots that instruct the processor should a spurious interrupt occur.

Once the 2732 is programmed, it is inserted into the EPROM slot on the right side of the programmer. The MC68701 to be programmed is inserted into the 40 pin socket on the left. Several plugs exist on the left side of the board. The red plug is +21 volts, the yellow plug is -12 volts and the black plug is ground. The DB-25 plug is the RS232 connection that interfaces the programmer to a terminal or terminal emulator on a personal computer.

The serial interface is running at 9600 baud, no start bits, eight data bits and one stop bit. Ensure that automatic linefeed is turned off at the terminal. (This keeps the screen aligned appropriately for easy monitoring of the copy programs function.) Set the blue DIP switch to 0111111. Turn on the power supplies and push the reset switch. The terminal first displays an opening logo. Upon entering the prompt to continue ('>') the MC68701 runs a blank check of it's internal EPROM. This blank check is actually a hex dump of all memory locations. If all locations are not $00, the EPROM requires further erasure. When the blank check is complete, the operator will be prompted to turn off DIP switch number three and turn on DIP switch number one. This gates 21 volts to the Vpp pin. When the character '>' is again pressed, a hexadecimal location counter will appear on the screen. It counts locations being programmed from $0000 to $07FF. (Remember the actual addresses being programmed are $F800 to $FFFF.) Upon completion of the programming, the terminal will give an audible alarm. At this point the operator turns off DIP switch number one and turns DIP switch number three back on. Programming is completed at this point, however, if the reset button is again pressed and the blank check function exercised, the EPROM can be verified as having been programmed. All instructions required to perform this function appear on the monitor as the process progresses.

The right hand three switches on the DIP switch set the mode of the processor. (Note from above that setting the switches to 111 is mode zero. The switch is aligned in an inverted binary pattern.) After the MC68701 is programmed, the processor can be changed to mode six by setting the switches to 0111010 and pushing the reset button. This will run the program in the MC68701. Depending on the software placed in the on-board EPROM, it may be capable of providing a mechanism of checking the program without having to install the microcomputer in another board.

APPENDIX C.

I. LABORATORY IV CURVE ACQUISITION

A. LABORATORY IV CURVE DEVICE

The hardware used for IV curve acquistion in the laboratory uses the exact equipment used in the NPS Solar Experiment. Although it is designed for 16 consecutive IV curves on 16 different cells, only one cell can be tested at a time due to the limitations of the KRATOS solar simulator.

B. LABORATORY IV CURVE SOFTWARE

CONVERT.ASM is the program segment of the solar experiment that actually acquires the IV curves for the sixteen cells in the experimental array. This is the same software used to acquire IV curves on-orbit with the exceptions that all interrupt and anneal functions are secured and it is written for direct interface with a terminal. This terminal interface includes instructions for the operation of the system.

```
•••••••••••••••••••••••••••••••••••••••••••••••••••••

  INCLUDE MACLIB.ASM      ;ENABLE PROGRAM TO READ MACRO
                          ;LIBRARY
  INCLUDE SYMDEF.ASM      ;AND SYMBOLIC DEFINITION
                          ;LIBRARY.
       DATA

•••••••••••••••••••••••••••••••••••••••••••••••••••••
PREPARE ADDRESSES FOR APPROPRIATE MESSAGES TO USER
•••••••••••••••••••••••••••••••••••••••••••••••••••••
```

```
                ORG         $FAF0       ;PLACE DATA IN ROM AT THIS
                                        ;LOCATION.

MSG1            BLKB                    25,$0A
                BYTE                    CR,’              SE RESEARCH’
                BYTE                    CR,LF,LF
                BYTE                    ’               16-CHANNEL’
                BYTE                    ’ SOLAR CELL ’
                BYTE                    ’IV CURVE TESTER’,CR,LF,LF
                BYTE                    ’                   LT GARY’
                BYTE                    ’SOMMERS, MAR’
                BYTE                    ’CH 1989’,CR,LF,LF,LF
                BYTE                    ’TURN OFF AUTOMATIC LINEFEED’
                BYTE                    ’ON TERMINAL!’
                BYTE                    CR,LF,LF,LF
                BYTE                    ’DO YOU WISH TO STEP THROUGH’
                BYTE                    ’PROCESS OR ALLOW IT TO RUN’
                BYTE                    ’AUTOMATICALLY?’,CR,LF
                BYTE                    ’(STEP THROUGH INITIALLY TO
                BYTE                    ’ADJUST THE SOLAR SIMULATOR’
                BYTE                    ’CONCENTRATION!)’,CR,LF
                BYTE                    ’(PRESS S FOR STEP OR A FOR’
                BYTE                    ’AUTOMATIC!)’,CR,LF,NUL

MSG2            BYTE                    CR,LF,’DO YOU REQUIRE’
                BYTE                    ’INSTRUCTIONS?’,CR,LF
                BYTE                    ’(Y OR N ONLY)’,CR,LF,NUL

MSG3            BYTE                    CR,LF,’INSTRUCTIONS:’,CR,LF
                BYTE                    ’ATTACH A VOLTMETER SET FOR
                BYTE                    ’MILLIVOLTS TO’
                BYTE                    ’ THE INPUT OF ADC #1.’,CR,LF
                BYTE                    ’ATTACH ANOTHER VOLTMETER SET’
                BYTE                    ’ SIMILARLY’
                BYTE                    ’ TO INPUT OF ADC #2.’,CR,LF
                BYTE                    ’IN IT”S PRESENT STATE (WITH’
                BYTE                    ’CELL #1 UNDER’
                BYTE                    ’ THE SOLAR SIMULATOR),’,CR,LF
                BYTE                    ’ADJUST THE CONCENTRATION BY’
                BYTE                    ’EITHER MOVING’
                BYTE                    ’ THE CELL UNDER THE’,CR,LF
                BYTE                    ’LIGHT OR BY ADJUSTING THE’
                BYTE                    ’ CURRENT TO THE’
                BYTE            ' LIGHT. (NO MORE THAN',CR,LF
                BYTE            '15 ON THE RELATIVE SCALE).'
                BYTE            '  IN MANY CASES'
                BYTE            ' YOU MUST DO BOTH.',CR,LF
                BYTE            'ONCE YOU HAVE ADJUSTED Voc,'
                BYTE            'STEP THROUGH'
                BYTE            ' WITH THE CHARACTER',CR,LF
                BYTE            '> UNTIL THE VOLTAGE AT ADC'
                BYTE            ' #1 JUST PASSES'
                BYTE            ' THROUGH 0 mv.  YOU',CR,LF
                BYTE            'ARE NOW AT Isc.  THE'
                BYTE            ' RESISTOR USED'
                BYTE            ' IN THE EMITTER IS',CR,LF
                BYTE            '30.1 OHMS.  YOU CAN READ THE'
                BYTE            ' VOLTAGE DIRECTLY'
                BYTE            ' FROM THE SECOND',CR,LF
                BYTE            'VOLTMETER AND THEN DIVIDE IT'
                BYTE            ' BY 30.1 TO GET CURRENT.'
                BYTE            '  ADJUST THE LIGHT TO ',CR,LF
                BYTE            'APPROPRIATE LEVELS AGAIN.'
                BYTE            '  THIS MAY REQUIRE'
                BYTE            ' SEVERAL ITERATIONS.'
                BYTE            CR,LF,NUL

MSG4            BYTE            CR,LF,'CONVERSION OF ALL CELLS'
                BYTE            'IS COMPLETE!',CR,LF,BEL,BEL
                BYTE            BEL,BEL,BEL,BEL,BEL,BEL,NUL

CRLF            BYTE            CR,LF,NUL


        DATA
••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
ADDITIONALLY, PUT PROGRAM STARTING ADDRESS IN THE FIRST
ADDRESS THAT THE PROCESSOR LOOKS AT UPON RESET.
••••••••••••••••••••••••••••••••••••••••••••••••••••••••••

        ORG     $FFFE

        WORD    $F800

        CODE
```

```
*****************************************************
COMMENCE PROGRAM AT ADDRESS SPECIFIED
*****************************************************

         ORG      $F800

         INIT               :MACRO TO INITIALIZE SERIAL
                            ;PORT TO 9600 BAUD,N,8,1,
                            ;ENABLE RAM AND SET PORT
                            ;FOUR TO OUTPUT ADDRESS A8
                            ;TO A15.

         LDS      #STACK    ;INITIALIZE STACK POINTER.

         TXSTR    MSG1      ;TRANSMIT OPENING MESSAGE.

         RXCHAR             ;RECEIVE CHARACTER FROM
                            ;CONSOLE.

         STAA     ANSWER    ;ANY CHARACTER OTHER THAN
                            ;'A' WILL STEP THROUGH
                            ;PROGRAM.

         CMPA     #'A'      ;IF THE ANSWER IS AUTOMATIC,
                            ;SKIP THE INSTRUCTIONS
         BEQ      CONV      ;MESSAGE.

         TXSTR    MSG2

         RXCHAR
         CMPA     #'N'
         BEQ      CONV

         TXSTR    MSG3

CONV:    TXSTR    CRLF
         LDX      #RMS      ;PUT RAM START LOCATION IN
                            ;INDEX REGISTER.


         LDAA     #$00      ;PUT FIRST CHANNEL NUMBER IN
                            ;ACCUMULATOR A.
```

```
        STAA    CHANNEL

OUTLP:  STAA    CHANNEL   ;SEND CHANNEL NUMBER TO
                          ;CHANNEL LATCH.
        LDAB    #$00      ;PUT FIRST D TO A COUNT IN
                          ;ACCUMULATOR B.

INLP:   STAB    DACON     ;SEND FIRST COUNT TO D TO A
                          ;FOR CONVERSION.

        PSHA              ;PUT ACC A ON THE STACK.

        LDAA    ANSWER    ;CHECK TO SEE IF AUTOMATIC
                          ;MODE.
        CMPA    #'A'
        BEQ     JUMP

        ECHO    '>'       ;WAIT FOR > TO STEP AGAIN.

JUMP:   PULA              ;RESTORE STACK.

        ATOD              ;MACRO TO OPERATE A TO D
                          ;CONVERTERS.
        CMPB    #255
        BEQ     NEXT
        INCB              ;INCREMENT THE COUNT.
        BRA     INLP

NEXT:   CMPA    #15       ;IS LAST CELL TESTED YET?
        BEQ     DONE      ;IF YES, BRANCH.

        INCA              ;IF NOT, INCREMENT THE
                          ;CHANNEL.

        BRA     OUTLP     ;REPEAT PROCESS.

DONE:   RAMDMP  RMS,RME   ;MACRO TO TRANSMIT RAM.

        TXSTR   MSG4      ;TRANSMIT END MESSAGE.

STOP1:  NOP
        BRA     STOP1
        END
```

C. OPERATING INSTRUCTIONS

The two plugs labelled 'CELL' is where the solar cell is plugged in. On the top of the tester, the red plug is for +15 volts, the black plug is for ground and the blue plug is for -12 volts. The RS232 cable is plugged into the 25 pin plug and connected to a terminal or terminal emulator. The serial characteristics are set to 9600 baud, no start bits, eight data bits and one stop bit. The two plugs on the top labelled 'CURRENT' and 'VOLTAGE' are direct connections to the inputs of the analog to digital converters. Voltmeters are connected to these with both instruments set to a zero to 500 millivolt scale. The KRATOS solar simulator is energized and 30 minutes is allowed for the light to stabilize. Place a calibration cell on the test block and adjust its temperature to match the temperature of its calibration IV curve. Turn on the power to the tester and press the reset switch. The program will give the option of either allowing the curve acquisition to occur automatically or it will allow the operator to step through each point one at a time. Select 'S' to adjust the solar simulator to the calibration cells curve. With 'S' selected, the cell is now at Voc. Adjust the solar simulator until the Voc of the calibration curve matches the cell Voc reading from the voltmeter attached to the 'VOLTAGE' plug. Once this has been done, step through the progrε until the voltage at the 'VOLTAGE' plug just passes through zero. This is Isc and the Isc read from the calibration curve must be matched to the Isc read by the voltmeter attached to the 'CURRENT' plug. (Remember here that the current is the voltage that is being read divided by 30.1.) This process to adjust the simulator will have to be repeated several times to ensure that the simulator is set accurately. Once the simulator is adjusted, remove the calibration cell and place the cell to be tested on the block. Reset the tester and select 'A' for automatic when prompted. The data acquired is downloaded to the terminal or terminal emulator. Since a terminal does not have the ability to store information, a terminal emulator is required if the data is to be used to plot graphs. Using a program such as PROCOMM, the data capture feature is used to acquire the data while it is being downloaded from the tester. This data is placed in a file called SOLAR.DAT. Store this file and in the same directory as the file PREPARE.EXE. (This program is provided in Appendix G.) Type the command PREPARE on the command line of the PC. This program converts the hexadecimal dump to floating point numbers and then adjusts the numbers to the voltage reference used on the digital to analog and analog to digital converters. A file is generated by this program called CELL.DAT. This file contains the adjusted, floating-point IV curve points in two columns. The first column corresponds to voltage and is the x-axis of the curve. The second column corresponds to current and is the y-axis of the curve. This file can be used by any quality graphics program to generate IV curves. Borland's QUATTRO was used effectively in the laboratory.

APPENDIX D.

I. SATELLITE EXPERIMENT PROGRAM

The satellite experiment software package consists of three discrete programs. MAIN.ASM is the core program which performs the IV curve acquisition as well as the anneal function. SATCLK.ASM is the interrupt handler for the on-board clock. RAMDUMP.ASM is the interrupt handler that dumps the acquired data through the serial port to the satellites main computer on demand of that computer.

```
• • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •
    INCLUDE MACLIB.ASM          ;ENABLE PROGRAM TO
                                ;READ MACRO LIBRARY
    INCLUDE SYMDEF.ASM          ;AND SYMBOLIC
                                ;DEFINITIONS
• • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •

            DATA
            ORG       $FFF0
            WORD      $F800     ;SPURIOUS INTERRUPT
                                ;RESTART VECTOR
            WORD      $F800     ;SPURIOUS INTERRUPT
                                ;RESTART VECTOR
            WORD      $FB00     ;CLOCK INTERRUPT
                                ;HANDLER
            WORD      $F800     ;SPURIOUS INTERRUPT
                                ;RESTART VECTOR
            WORD      $FD00     ;RAMDUMP INTERRUPT
                                ;HANDLER
            WORD      $F800     ;SPURIOUS INTERRUPT
                                ;RESTART VECTOR
            WORD      $F800     ;SPURIOUS INTERRUPT
                                ;RESTART VECTOR
```

```
         WORD     $F800     ;MAIN PROGRAM START
                            ;ADDRESS
         CODE

         ORG      $F800

MAIN:    INIT               ;MACRO TO INITIALIZE SERIAL
                            ;PORT 9600 BAUD,N,8,1,
                            ;ENABLE RAM AND SET PORT FOUR
                            ;TO OUTPUT ADDRESSES.
         LDS      #STACK    ;INITIALIZE STACK POINTER.

         LDAA     #$00      ;INITIALIZE CLOCK TO ZERO.
         STAA     SECOND
         STAA     MINUTE
         STAA     HOUR
         STAA     DAYANN    ;ZERO ANNEAL DAY COUNTER
         STAA     DAYIV     ;AND IV CURVE DAY COUNTER.
         STAA     TOTAL

         LDAA     #240      ;LOAD DOWN COUNTER FOR SERIAL
                            ;TEST DATA DUMP.
TEST:    TXCHAR   #$FF      ;TRANSMIT $FF.
         DELAY    250       ;WAIT FOR 250 MILLISECONDS.
         DECA               ;DECREMENT THE COUNTER BY ONE.
         CMPA     #$00      ;IS THE COUNTER AT ZERO?
         BNE      TEST      ;IF NOT, CONTINUE DATA DUMP.

         CLI                ;CLEAR INTERRUPT BIT IN CCR.
         LDAA     #$08      ;ENABLE OUTPUT COMPARE FLAG
         STAA     TCSR      ;BY WRITING TO TCSR.

LOOP:    LDAA     DAYANN    ;LOAD THE ANNEAL DAY COUNTER.
         CMPA     #$B3      ;IS IT THE 180TH DAY?
         BNE      NEXT      ;IF NOT, GO TO IV CURVE
                            ;SEGMENT.
         JMP      ANNL      ;IF YES, JUMP TO ANNEAL
                            ;SUBROUTINE.

NEXT:    LDAA     DAYIV     ;LOAD THE IV CURVE DAY
                            ;COUNTER.
```

```
          CMPA      #$01        ;IS IT THE TENTH DAY AFTER
                                ;ROLLOVER OF IV DAY COUNTER?

          BNE       LOOP        ;IF NOT, GO BACK TO PROGRAM
                                ;START.
          LDAA      HOUR        ;IS IT TIME 00:00:00?
          CMPA      #$00        ;IF NOT, RETURN TO PROGRAM
                                ;START.
          BNE       LOOP
          LDAA      MINUTE
          CMPA      #$00
          BNE       LOOP
          LDAA      SECOND
          CMPA      #$00
          BNE       LOOP
          JMP       CONV        ;IF YES, JUMP TO IV CURVE
                                ;SUBROUTINE.

..............................................................
ANNEAL SUBROUTINE
..............................................................

ANNL:     PSHX                  ;SAVE ALL REGISTERS ON THE
          PSHA                  ;STACK.
          PSHB

          LDAA      #$10        ;PUT FIRST CELL NUMBER IN
                                ;ACCUMULATOR A
                                ;AND SEND 15 VOLTS TO CELL.

ALOOP:    STAA      ANLATCH     ;SEND CHANNEL NUMBER TO
                                ;ANNEAL LATCH.
          PSHA                  ;SAVE THE CHANNEL NUMBER.

TIME:     LDAA      HOUR        ;GET THE CURRENT CLOCK VALUE
                                ;AND PLACE IT IN TEMPORARY
          STAA      TEMPA       ;STORAGE FOR LATER COMPARISON.
          LDAA      MINUTE
          STAA      TEMPB
          LDAA      SECOND
          STAA      TEMPC
```

```
        DELAY    250       ;DELAY ONE SECOND TO ALLOW
                           ;CLOCK TO ADVANCE SO THAT
        DELAY    250       ;ANNEALING DOES NOT STOP
        DELAY    250       ;IMMEDIATELY UPON THE
        DELAY    250       ;FOLLOWING COMPARISON.

LOOP1:  LDAA     SECOND    ;THIS COMPARISON CHECKS TO
                           ;ENSURE THAT 24 HOURS HAVE
                           ;ELAPSED.
        LDAB     TEMPC
        CBA
        BNE      LOOP1
        LDAA     MINUTE
        LDAB     TEMPB
        CBA
        BNE      LOOP1
        LDAA     HOUR
        LDAB     TEMPA
        CBA
        BNE      LOOP1

        PULA               ;RESTORE THE CHANNEL NUMBER.

        CMPA     #$1F      ;HAS THE LAST CELL BEEN
                           ;ANNEALED?

        BEQ      QUIT      ;IF YES, QUIT.
        INCA               ;IF NOT, INCREMENT TO THE NEXT
        JMP      ALOOP     ;AND START TIMING SEQUENCE
                           ;AGAIN.


QUIT:   LDAA     #$00      ;TURN OFF ALL POWER TO ANNEAL
                           ;CIRCUITRY.
        STAA     ANLATCH

STOP2:  PULB               ;RESTORE REGISTERS.
        PULA
        PULX
        JMP      NEXT      ;RETURN FROM SUBROUTINE.
```

```
• • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •
IV CURVE ACQUISITION SUBROUTINE
• • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •

CONV:   PSHX              ;SAVE ALL REGISTERS.
        PSHA
        PSHB
        LDX     #RMS      ;PUT RAM START LOCATION IN
                          ;INDEX REGISTER.
        LDAA    #$00      ;PUT FIRST CHANNEL NUMBER IN
                          ;ACCUMULATOR A.
        STAA    CHANNEL

OUTLP:  STAA    CHANNEL   ;SEND CHANNEL NUMBER TO
                          ;CHANNEL LATCH.
        LDAB    #$00      ;PUT FIRST D TO A COUNT IN
                          ;ACCUMULATOR B.

INLP:   STAB    DACON     ;SEND FIRST COUNT TO D TO A
                          ;FOR CONVERSION.
        ATOD              ;MACRO TO OPERATE A TO D
                          ;CONVERTERS.
        CMPB    #255
        BEQ     NEXT1
        INCB              ;INCREMENT THE COUNT.
        BRA     INLP

NEXT1:  CMPA    #15       ;IS LAST CELL TESTED YET?
        BEQ     STOP1     ;IF YES, BRANCH OUT OF PROGRAM.
        INCA              ;IF NOT, INCREMENT THE CHANNEL
        BRA     OUTLP     ;AND REPEAT PROCESS.

STOP1:  PULB              ;RESTORE ALL REGISTERS.
        PULA
        PULX
        JMP     LOOP      ;RETURN FROM SUBROUTINE.

STOP3:  NOP               ;PROGRAM IS ENDLESS SO THIS
                          ;LINE IS NEVER REACHED.
        JMP     MAIN      ;IF IT DOES, RESTART.
        END
```

```
•••••••••••••••••••••••••••••••••••••••••••••••••
```

SATCLK.ASM is the interrupt handler (IRQ2) for experiment time monitoring. IRQ2 is an internal interrupt line that is generated by the on-board timer output compare register reaching a specified value. The clock that is generated keeps track of days up to 180 days for the anneal clock and ten days for the IV curve clock. Each of these clocks rolls over to zero after their maximum value has been reached.

```
•••••••••••••••••••••••••••••••••••••••••••••••••
    INCLUDE MACLIB.ASM          ;MAKE MACROS AVAILABLE TO
                                ;PROGRAM.
    INCLUDE SYMDEF.ASM          ;MAKE SYMBOL DEFINITIONS
                                ;AVAILABLE.
•••••••••••••••••••••••••••••••••••••••••••••••••

        CODE

        ORG       $FB00         ;ADDRESS OF PROGRAM IN ROM.

        LDAA      TCSR          ;READ TIMER CONTROL STATUS
                                ;REGISTER TO CLEAR INTERRUPT
                                ;BIT.
        LDAA      #$EF          ;LOAD THE OUTPUT COMPARE
                                ;REGISTERS
        STAA      OCRM          ;WITH $EFCE WHICH, REPRESENTS
                                ;CLOCK CYCLES REQUIRED FOR
        LDAA      #$CE          ;50 MS MINUS SEVEN CLOCK CYCLES.
        STAA      OCRL
        STAA      CRM           ;PRESET COUNTER REGISTER TO
                                ;$FFF8.

        LDAA      TOTAL         ;TAKE VALUE IN TOTAL (THE TOTAL
                                ;NUMBER OF 50 MS SEGMENTS THAT
                                ;HAVE ELAPSED SINCE LAST
                                ;INTERRUPT) AND,

        INCA                    ;INCREMENT THE COUNT.
        STAA    TOTAL    ;STORE THE NEW COUNT BACK IN
                         ;ACCUMULATOR A.
        CMPA    #$14     ;IF 20 HAVE OCCURRED, UPDATE
                         ;THE CLOCK.
        BEQ     BEGIN

        RTI              ;IF NOT RETURN AND WAIT FOR THE
                         ;NEXT INTERRUPT.

..........................................................
CLOCK UPDATE SEGMENT
..........................................................

BEGIN:  LDAA    #00
        STAA    TOTAL    ;ZERO THE TOTAL NUMBER OF
                         ;ELAPSED 50 MS SEGMENTS.

        LDAA    SECOND   ;GET THE LAST VALUE OF SECOND.
        INCA             ;INCREMENT THE SECOND
        STAA    SECOND   ;AND STORE IT BACK IN
                         ;MEMORY.
        CMPA    #$3B     ;IF ITS GREATER THAN 59,
                         ;ZERO IT
        BGT     MIN      ;AND GO TO MINUTES.
        JMP     SEND     ;OTHERWISE QUIT AND RETURN.

MIN:    LDAA    #0
        STAA    SECOND

        LDAA    MINUTE
        INCA
        STAA    MINUTE
        CMPA    #$3B
        BGT     HR
        JMP     SEND

HR:     LDAA    #0       ;TO INCREMENT HOURS. ZERO
                         ;THE MINUTES AND INCREMENT
        STAA    MINUTE   ;THE HOUR.
        LDAA    HOUR
        INCA
        STAA    HOUR
        CMPA    #$17     ;IF HOUR IS GREATER THAN 23
        BGT     DYAN     ;GO TO ANNEAL DAY COUNTER,
        BRA     SEND     ;ELSE, RETURN.

DYAN:   LDAA    #00      ;ZERO THE HOURS.
        STAA    HOUR
        LDAA    DAYANN   ;INCREMENT THE ANNEAL DAY
                         ;COUNTER.
        INCA
        STAA    DAYANN
        CMPA    #$B3     ;IF IT'S NOT THE 180TH DAY
        BLE     DYIV     ;GO TO IV CURVE COUNTER.
        LDAA    #$00     ;IF IT IS, ZERO THE COUNTER.
        STAA    DAYANN

DYIV:   LDAA    DAYIV    ;INCREMENT THE IV DAY COUNTER.
        INCA
        STAA    DAYIV
        CMPA    #$09     ;IF IT'S NOT THE TENTH DAY
        BLE     SEND     ;RETURN.

        LDAA    #$00     ;IF IT IS, ZERO THE DAY
                         ;COUNTER,
        STAA    DAYIV

SEND:   RTI              ;RETURN FROM INTERRUPT
        END

............................................................
```

RAMDUMP.ASM is the interrupt handler that downloads the contents of RAM through the serial port to the satellite's main on-board computer for transmission to the ground. For this to occur, IRQ1 (pin five) must be driven low for at least a single clock cycle.

```
............................................................
   INCLUDE MACLIB.ASM    ;MAKE MACROS AVAILABLE
                         ;TO PROGRAM
   INCLUDE SYMDEF.ASM    ;MAKE SYMBOLIC
                         ;DEFINITIONS AVAILABLE
............................................................
```

```
          CODE

          ORG       $FD00       ;START LOCATION IN ROM.

RAMDP:    LDX       #RMS        ;LOAD IDX WITH RAMSTART
                                ;LOCATION.

LOOPA:    LDAA      X           ;TAKE BYTE OF DATA.
          TAB                   ;ACC A AND ACC B HAS DATA.
          ANDA      #$F0        ;MASK ACCUMULATORS.
          ANDB      #$0F
          LSRA                  ;SHIFT TO LOWER BYTE.
          LSRA
          LSRA
          LSRA
          ADDA      #$30        ;ADD $30 TO BOTH.
          ADDB      #$30
          CMPA      #$3A        ;JUMP CHARACTER $3A TO $40.
          BGE       ADJ

CONT:     CMPB      #$3A
          BGE       ADJ2

NEXT1:    TRE                   ;IS TRANSMIT REGISTER EMPTY?
          STAA      TDR         ;TRANSMIT LOW HEX CHARACTER.
          TRE                   ;IS TRANSMIT REGISTER EMPTY?
          STAB      TDR         ;TRANSMIT HIGH HEX CHARACTER.
          CPX       #RME        ;HAS ALL DATA BEEN CONVERTED?
          BEQ       STOP
          INX                   ;INCREMENT TO NEXT CHARACTER.
          BRA       LOOPA

ADJ:      ADDA      #7          ;ADD SEVEN TO EACH IF NUMBER IS
                                ;$40 OR GREATER.
          BRA       CONT

ADJ2:     ADDB      #7
          BRA       NEXT1

STOP:     RTI                   ;RETURN FROM INTERRUPT.
          END
```

APPENDIX E.

I. MACRO DEFINITION LIBRARY

MACLIB.ASM is the library of macros that is used by the on-orbit solar cell experiment. This library contains not only macros for the controlling of IV and anneal functions, but also, processor initiation and ground terminal interface functions. It is accessed by the main program by making it an 'INCLUDE' file in the main program's header.

```
•••••••••••••••••••••••••••••••••••••••••••••••••••••••••
TXCHAR TRANSMITS A SINGLE CHARACTER TO THE SERIAL PORT
TRANSMISSION REGISTER (TDR).
•••••••••••••••••••••••••••••••••••••••••••••••••••••••••

TXCHAR     MACRO          CHAR

       PSHA               ;MAKE MACRO TRANSPARENT.
       LDAA     CHAR      ;PUT BYTE TO TRANSMIT IN ACC A.
       TRE                ;IS TRANSMIT REGISTER EMPTY?
       STAA     TDR       ;IF YES, SEND BYTE TO TRANSMIT
                          ;DATA REGISTER.
       PULA
       ENDM

•••••••••••••••••••••••••••••••••••••••••••••••••••••••••
TXSTR TRANSMITS A STRING OF CHARACTERS TO THE SERIAL PORT
TRANSMISSION  REGISTER  (TDR).
•••••••••••••••••••••••••••••••••••••••     •••••••••••••

TXSTR      MACRO    ADDRST

       PSHA               ;MAKE MACRO TRANSPARENT.
       PSHX
```

```
          LDX      #ADDRST  ;PUT ADDRESS OF STRING IN INDEX
                            ;REGISTER.

LOOP#:    LDAA     X        ;PUT CONTENTS OF INDEX POINTER
                            ;IN ACCUMULATOR A.
          CMPA     #NUL     ;STOP TRANSMISSION AT END OF
                            ;STRING.
          BEQ      FINIS#

          PSHA              ;NEXT FIVE LINES CHECK TO ENSURE
                            ;TRANSMIT DATA REGISTER IS
          LDAA     #$20     ;EMPTY.
WAIT4#:   BITA     TRCSR
          BEQ      WAIT4#
          PULA
          STAA     TDR      ;PUT BYTE IN TRANSMIT DATA
                            ;REGISTER.
          INX               ;INCREMENT THE INDEX (ADDRESS
                            ;OF BYTE).
          BRA      LOOP#    ;GET THE NEXT BYTE TO TRANSMIT.

FINIS#:   PULX
          PULA
          ENDM

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
RXCHAR RECEIVES A SINGLE CHARACTER FROM THE SERIAL PORT
RECEIVE DATA REGISTER. (RDR). IT THEN TRANSMITS IT BACK TO
THE SCREEN.
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

RXCHAR      MACRO

          RRF               ;IS RECEIVE REGISTER FULL?
          LDAA     RDR      ;GET BYTE FROM RECEIVE DATA
                            ;REGISTER.
          TRE               ;IS TRANSMIT REGISTER EMPTY?
          STAA     TDR      ;ECHO IT BACK TO TERMINAL.
          ENDM

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
ECHO RECEIVES A SINGLE CHARACTER AND THEN COMPARES IT TO
A SET VALUE TO START SOME SEQUENCE.  IT TRANSMITS IT BACK TO
```

THE SCREEN THROUGH THE IMBEDDED MACRO RXCHAR AND TXCHAR.

```
•••••••••••••••••••••••••••••••••••••••••••••••••••••••••
ECHO        MACRO   CHAR

            PSHA

LOOP#:      RXCHAR              ;KEYBOARD INTERFACE SEGMENT.
            CMPA      #CHAR     ;HAS THE APPROPRIATE KEY BEEN
                                ;PUSHED?
            BNE       LOOP#     ;IF NOT, THEN ECHO CHARACTERS.
            PULA
            ENDM

•••••••••••••••••••••••••••••••••••••••••••••••••••••••••
```

INIT ENABLES THE ONBOARD RAM, ALIGNS PORT FOUR TO BE ADDRESS OUTPUTS A8-A15, SETS THE RATE AND MODE OF THE SERIAL PORT TO 9600,N,8,1 AND SETS THE SERIAL PORT TO BOTH TRANSMIT AND RECEIVE.

```
•••••••••••••••••••••••••••••••••••••••••••••••••••••••••

INIT        MACRO

            LDAA      #$FF      ;ENABLE ONBOARD RAM AND SET
                                ;PORT FOUR TO BECOME ADDRESS
            STAA      RAMCR     ;OUTPUTS A8-A15.
            STAA      P4DDR
            LDAA      #$05      ;SET RATE AND MODE FOR SERIAL
                                ;TRANSMISSION.
            STAA      RMCR
            LDAA      TRCSR     ;DUMMY READ TRC REGISTER.
            LDAA      #$0A      ;SET PORT FOR TRANSMIT AND
                                ;RECEIVE.
            STAA      TRCSR
            ENDM

•••••••••••••••••••••••••••••••••••••••••••••••••••••••••
```

RRF (RECEIVE REGISTER FULL) IS A CHECK OF THE RECEIVE REGISTER TO ENSURE THAT IT IS FULL PRIOR TO THE PROCESSOR ATTEMPTING TO READ IT FOR VALID DATA.

```
•••••••••••••••••••••••••••••••••••••••••••••••••••••••••
```

```
RRF           MACRO

        PSHA                  ;MAKE MACRO TRANSPARENT.
        LDAA     #$80         ;PUT BINARY 1000 0000 INTO
                              ;ACCUMULATOR A.

WAIT1#: BITA     TRCSR        ;'AND' IT TO TRCSR WITHOUT
                              ;CHANGING EITHER.
        BEQ      WAIT1#       ;BRANCH TO WAIT1 IF NOT FULL.
        PULA
        ENDM

•••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
TRE (TRANSMIT REGISTER EMPTY) IS A CHECK OF THE TRANSMIT
REGISTER TO ENSURE THAT IT IS EMPTY PRIOR TO THE PROCESSOR
ATTEMPTING TO WRITE DATA TO IT.
•••••••••••••••••••••••••••••••••••••••••••••••••••••••••••

TRE           MACRO

        PSHA                  ;MAKE MACRO TRANSPARENT.
        LDAA     #$20         ;PUT BINARY 0010 0000 INTO
                              ;ACCUMULATOR A.

WAIT2#: BITA     TRCSR        ;AND IT TO TRCSR WITHOUT
                              ;CHANGING EITHER.
        BEQ      WAIT2#       ;BRANCH IF FULL TO WAIT2.
        PULA
        ENDM

•••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
DELAY PROVIDES A DELAY OF UP TO 255 MILLISECONDS DEPENDING
ON THE VALUE OF THE ARGUMENT USED.
•••••••••••••••••••••••••••••••••••••••••••••••••••••••••••

DELAY         MACRO    LENGTH

        PSHA
        PSHB
        LDAB     #LENGTH      ;HOW MANY MILLISECONDS?

DELY2#: LDAA     #204         ;FIRST LOOP COUNTS DOWN FOR A
                              ;ONE MS DELAY.
```

```
DELY1#:DECA
       BNE     DELY1#
       DECB               ;SECOND LOOP COUNTS NR OF MS'S.
       BNE     DELY2#
       PULB
       PULA
       ENDM

•••••••••••••••••••••••••••••••••••••••••••••••••••••••
WAIT PROVIDES A DELAY OF UP TO 255, 25 MICROSECOND SEGMENTS
DEPENDING ON THE VALUE OF THE ARGUMENT USED.
•••••••••••••••••••••••••••••••••••••••••••••••••••••••

WAIT          MACRO    LENGTH

       PSHA
       PSHB
       LDAB    #LENGTH    ;HOW MANY 25 MICROSECOND
                          ;SEGMENTS?

WAIT4#: LDAA   #7         ;FIRST LOOP COUNTS DOWN FOR 25
                          ;MICROSECOND DELAY.

WAIT3#: DECA
       BNE     WAIT3#
       DECB               ;SECOND LOOP COUNTS NUMBER OF
                          ;SEGMENTS
       BNE     WAIT4#
       PULB
       PULA
       ENDM

•••••••••••••••••••••••••••••••••••••••••••••••••••••••
RAMDMP READS AN AREA OF MEMORY FROM ARGUMENTS ST TO
FIN AND DUMPS IT TO THE SERIAL PORT AFTER CONVERTING THE
RAM CONTENTS TO HEXADECIMAL.
•••••••••••••••••••••••••••••••••••••••••••••••••••••••

RAMDMP        MACRO    ST,FIN

       LDX     #ST        ;LOAD IDX WITH RAMSTART
                          ;LOCATION.
```

```
LOOP#:  LDAA    X       ;TAKE BYTE OF DATA.
        TAB             ;ACC A AND ACC B HAS DATA.
        ANDA    #$F0    ;MASK ACCUMULATORS.
        ANDB    #$0F
        LSRA            ;SHIFT TO LOWER BYTE.
        LSRA
        LSRA
        LSRA
        ADDA    #$30    ;ADD $30 TO BOTH.
        ADDB    #$30
        CMPA    #$3A    ;JUMP CHARACTER $3A TO $40.
        BGE     ADJ#

CONT#:  CMPB    #$3A
        BGE     ADJ2#

NEXT1#:TRE
        STAA    TDR
        TRE
        STAB    TDR
        CPX     #FIN    ;HAS ALL DATA BEEN CONVERTED?
        BEQ     STOP#
        INX
        BRA     LOOP#

ADJ#:   ADDA    #7      ;ADD SEVEN TO EACH IF NUMBER IS
                        ;$40 OR GREATER
        BRA     CONT#

ADJ2#:  ADDB    #7
        BRA     NEXT1#

STOP#:  NOP
        ENDM
```

••

RAMDMP1 READS AN AREA OF MEMORY FROM ARGUMENTS ST TO FIN AND DUMPS IT TO THE SERIAL PORT AFTER CONVERTING THE RAM CONTENTS TO HEXADECIMAL. THIS IS THE SAME MACRO AS RAMDMP EXCEPT THAT IT PAUSES BETWEEN FULL SCREENS FOR A MAXIMUM OF THREE SCREENS.

••

```
RAMDMP1     MACRO     ST,FIN

            LDX       #ST          ;LOAD IDX WITH RAMSTART
                                   ;LOCATION.

LOOP#:      LDAA      X            ;TAKE BYTE OF DATA.
            TAB                    ;ACC A AND ACC B HAS DATA.
            ANDA      #$F0         ;MASK ACCUMULATORS.
            ANDB      #$0F
            LSRA                   ;SHIFT TO LOWER BYTE.
            LSRA
            LSRA
            LSRA
            ADDA      #$30         ;ADD $30 TO BOTH.
            ADDB      #$30
            CMPA      #$3A         ;JUMP CHARACTER $3A TO $40.
            BGE       ADJ#

CONT#:      CMPB      #$3A
            BGE       ADJ2#

NEXT1#:TRE
            STAA      TDR
            TRE
            STAB      TDR
            CPX       #ST+$347     ;STOP THE SCREEN SCROLL IF
                                   ;SCREEN IS FULL.
            BEQ       MESS#
            CPX       #ST+$68F     ;STOP THE SCREEN SCROLL IF
                                   ;SCREEN IS FULL.
            BEQ       MESS#

CONT1#:CPX            #FIN         ;HAS ALL DATA BEEN CONVERTED?
            BEQ       STOP#
            INX
            BRA       LOOP#

ADJ#:       ADDA      #7           ;ADD SEVEN TO EACH IF NUMBER IS
                                   ;$40 OR GREATER
            BRA       CONT#

ADJ2#:      ADDB      #7
            BRA       NEXT1#
```

```
MESS#:  TXSTR    WAITMSG  ;MESSAGE THAT SCREEN IS FULL.
        ECHO     '>'      ;ARROW MUST BE PRESSED TO
        TXCHAR   #CR      ;CONTINUE.
        TXCHAR   #LF
        BRA      CONT1#

STOP#:  NOP
        ENDM

• • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •
ATOD STARTS THE A TO D CONVERTERS, READS THE OUTPUT OF
THE CONVERTERS AND THEN STORES THE RESULTS IN
CONSECUTIVE BYTES OF MEMORY. INDEX REGISTER HAS RAM
ADDRESS ON ENTRY.
• • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •

ATOD         MACRO

        PSHA              ;MAKE TRANSPARENT.
        STAA     START    ;DUMMY WRITE ADC'S TO START
                          ;CONVERSION.
        WAIT     4        ;DELAY 100 MICROSECONDS.
        LDAA     OEADC1   ;OUTPUT ENABLE AND READ
                          ;VOLTAGE.
        STAA     X        ;PUT VOLTAGE RESULTS
                          ;IN RAM

        INX               ;INCREMENT RAM LOCATION.
        LDAA     OEADC2   ;OUTPUT ENABLE AND READ
                          ;CURRENT.
        STAA     X        ;PUT CURRENT RESULT IN RAM
                          ;LOCATION.
        INX               ;INCREMENT RAM LOCATION.
        PULA              ;RESTORE STACK.
        ENDM

• • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •
RAMDMP2 DUMPS AN AREA OF MEMORY TO THE SERIAL PORT
WITHOUT THE CHANGE TO HEXADECIMAL AS IN THE MACRO
RAMDMP.
• • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •

RAMDMP2      MACRO     ST,FIN
```

```
          PSHA
          LDX       #ST         ;LOAD IDX WITH RAMSTART
                                ;LOCATION.

LOOP#:    TXCHAR    X           ;TRANSMIT CONTENTS OF X.
          CPX       #FIN        ;IS IT LAST LOCATION?
          BEQ       STOP#       ;IF YES, EXIT MACRO.
          INX                   ;IF NOT, INCREMENT LOCATION
          BRA       LOOP#       ;AND GET ANOTHER.

STOP#:    PULA
          ENDM

************************************************************
DUMP EMPTIES AN SINGLE BYTE OF MEMORY TO THE SERIAL PORT.
THIS IS USED AS A DECIMAL DUMP WHEN USING ONLY ASCII
CHARACTERS ZERO THROUGH NINE IN A COUNTER.
************************************************************

DUMP         MACRO     VALUE

          PSHA                  ;MAKE TRANSPARENT.
          LDAA      VALUE       ;LOAD ACC A WITH BYTE TO BE
                                ;DUMPED.
          ADDA      #$30        ;ADD $30 TO CORRECT FOR ASCII.
          TRE                   ;IS TRANSMIT REGISTER EMPTY?
          STAA      TDR         ;IS YES, TRANSMIT DATA.
          PULA
          ENDM

************************************************************

          END
```

APPENDIX F.

I. SYMBOL DEFINITION LIBRARY

SYMDEF.ASM is the symbol definition library for the NPS Solar Experiment. This library is accessed by the main program with the 'INCLUDE' statement in the main program's header.

```
•••••••••••••••••••••••••••••••••••••••••••••••••••••••

P1DDR:      EQU  $00      ;PORT ONE DATA DIRECTION
                          ;REGISTER
P2DDR:      EQU  $01      ;PORT TWO DATA DIRECTION
                          ;REGISTER
P1DR:       EQU  $02      ;PORT ONE DATA REGISTER
P2DR:       EQU  $03      ;PORT TWO DATA REGISTER
P3DDR:      EQU  $04      ;PORT THREE DATA DIRECTION
                          ;REGISTER
P4DDR:      EQU  $05      ;PORT FOUR DATA DIRECTION
                          ;REGISTER
P3DR:       EQU  $06      ;PORT THREE DATA REGISTER
P4DR:       EQU  $07      ;PORT FOUR DATA REGISTER
TCSR:       EQU  $08      ;TIMER CONTROL AND STATUS
                          ;REGISTER
CRM:        EQU  $09      ;COUNTER REGISTER MOST
                          ;SIGNIFICANT
CRL:        EQU  $0A      ;COUNTER REGISTER LEAST
                          ;SIGNIFICANT
OCRM:       EQU  $0B      ;OUTPUT COMPARE REGISTER MOST
                          ;SIGNIFICANT
OCRL:       EQU  $0C      ;OUTPUT COMPARE REGISTER LEAST
                          ;SIGNIFICANT
ICRM:       EQU  $0D      ;INPUT CAPTURE REGISTER MOST
                          ;SIGNIFICANT
ICRL:       EQU  $0E      ;INPUT CAPTURE REGISTER LEAST
                          ;SIGNIFICANT
RMCR:       EQU  $10      ;RATE AND MODE CONTROL REGISTER
```

```
TRCSR:      EQU  $11      ;TRANS/RECEIVE CONTROL STATUS
                          ;REGISTER
RDR:        EQU  $12      ;RECEIVE DATA REGISTER
TDR:        EQU  $13      ;TRANSMIT DATA REGISTER
RAMCR:      EQU  $14      ;RAM CONTROL REGISTER

•••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
ADDRESS DEFINITIONS FOR SOLAR PROJECT HARDWARE
•••••••••••••••••••••••••••••••••••••••••••••••••••••••••••

STACK:      EQU  $00F0    ;STACK START LOCATION IN RAM
SAVER:      EQU  $00A0    ;ROM ADDRESS POINTER
SAVEEP:     EQU  $00A2    ;EPROM ADDRESS POINTER
ANSWER:     EQU  $00A4    ;STORAGE FOR INITIAL QUESTION IN
                          ;PROGRAM
SRA:        EQU  $00A8    ;STORAGE FOR PROGRAMMER SCREEN
                          ;COUNTER
SRB:        EQU  $00A7    ;STORAGE FOR PROGRAMMER SCREEN
                          ;COUNTER
WAIT:       EQU  $EFBF    ;COUNT FOR 50MS DELAY
EPRST:      EQU  $F800    ;INTERNAL EPROM START LOCATION
EPREND:     EQU  $FFFF    ;INTERNAL EPROM END LOCATION
ROMST:      EQU  $4800    ;COPY ROM START LOCATION
ROMEND:     EQU  $4FFF    ;COPY ROM END LOCATION
RMS:        EQU  $4000    ;RAM START ADDRESS
RME:        EQU  $5FFF    ;RAM END ADDRESS
RAM2ST:     EQU  $8000    ;64K RAM START ADDRESS
RAM2END:    EQU  $8FFF    ;64K RAM END ADDRESS
CROM:       EQU  $B000    ;COPY ROM START ADDRESS
CEND:       EQU  $BFFF    ;COPY ROM END ADDRESS
ANNEAL:     EQU  $1000    ;CHANNEL LATCH FOR ANNEAL
                          ;MULTIPLEXER
START:      EQU  $2000    ;ADC START AND ADDRESS LATCH
FOR
                          ;BOTH ADC'S
DACON:      EQU  $6000    ;ADDRESS OF DATA LATCH FOR
                          ;D TO A CONVERTER
CHANNEL:    EQU  $8000    ;CHANNEL SELECT ADDRESS LATCH
OEADC2:     EQU  $A000    ;OUTPUT ENABLE FOR A TO D
                          ;CONVERTER #2
OEADC1:     EQU  $C000    ;OUTPUT ENABLE FOR A TO D
                          ;CONVERTER #1
```

```
• • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •
MEMORY STORAGE LOCATIONS FOR THE CLOCK. THESE ARE IN THE
MICROCOMPUTER'S INTERNAL RAM.
• • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •
SECOND:      EQU  $00B0     ;SECOND DIGIT
MINUTE:      EQU  $00B1     ;MINUTE DIGIT
HOUR:        EQU  $00B2     ;HOUR DIGIT
DAYANN:      EQU  $00B3     ;ANNEAL DAY COUNTER
DAYIV:       EQU  $00B4     ;IV CURVE DAY COUNTER
TEMPA:       EQU  $00B5     ;TEMPORARY SECOND STORAGE
TEMPB:       EQU  $00B6     ;TEMPORARY MINUTE STORAGE
TEMPC:       EQU  $00B7     ;TEMPORARY HOUR STORAGE
TOTAL:       EQU  $00B8     ;LOCATION TO STORE TOTAL NUMBER
                            ;OF 50 MILLISECOND CYCLES THAT
                            ;HAVE OCCURRED.

• • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •
TERMINAL CONTROL CHARACTERS
• • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •

CR:          EQU  $0D       ;CARRIAGE RETURN
BLANK:       EQU  $20       ;BLANK CHARACTER
LF:          EQU  $0A       ;TERMINAL LINEFEED
BEL:         EQU  $07       ;BELL
NUL:         EQU  $00       ;END OF STRING CHARACTER
ESC:         EQU  $1B       ;ESCAPE CHARACTER

• • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •
END
```

APPENDIX G.

I. GROUND STATION PROGRAM

The ground station program PREPARE.EXE is a ‘FORTRAN’ program that takes a remote file of the hexadecimal dump from the satellite IV curve device and converts it to floating point decimal. All of the mathematical manipulation of the satellite data is done in this program. The strategy of doing all mathematics away from the satellite minimizes the amount of ROM space required. No curve smoothing is performed in this program. This program must be run individually on each cell.

```
*........................................................
$NOFLOATCALLS
*........................................................

      DIMENSION J(1:512)
      DIMENSION X(1:256),Y(1:256),Z(1:512)

*........................................................
*     J IS THE HEXADECIMAL INPUT STRING, Z IS THE FLOATING
*     POINT REPRESENTATION OF THE INPUT STRING AND X AND Y
*     ARE THE OUTPUT STRINGS
*........................................................

*........................................................
*     OPEN FILES REQUIRED TO ACQUIRE AND STORE DATA
*........................................................

      OPEN (7,FILE='SOLAR.DAT',STATUS='OLD')
      OPEN (6,FILE='CELL.DAT',STATUS='UNKNOWN')
```

```
*.....................................................
*    CONSTANT VALUES ARE DETERMINED BY THE VOLTAGE
*    REFERENCES USED ON THE TWO A TO D CONVERTERS AND
*    THE D TO A CONVERTER.
*.....................................................

     CONST=2.058/256.

*.....................................................
*    RESISTOR VALUE IS IN K OHMS
*.....................................................

     RESISTOR=.01005

*.....................................................
*    READ IN DATA FROM A DISK FILE ON THE DEFAULT DRIVE
*    CALLED CELL.DAT.  THE DATA IS THE HEXADECIMAL
*    REPRESENTATION OF 512 BYTES IN A CONTINUOUS STREAM.
*    THE HEXADECIMAL BYTES ARE CONVERTED TO FLOATING
*    POINT DECIMAL FOR FURTHER ALGEBRAIC MANIPULATION.
*.....................................................

       READ (7,'(512Z2)') (J(N),N=1,512)
     DO 1 N=1,512,1
       Z(N)=FLOAT(J(N))
1    CONTINUE

*.....................................................
*    CALCULATE THE VOLTAGES IN MILLIVOLTS
*.....................................................

       K=1
     DO 2 N=1,511,2
       X(K)=(Z(N)*CONST*1000.)
       K=K+1
2    CONTINUE

*.....................................................
*    CALCULATE THE CURRENTS ACROSS THE RESISTORS IN
*    MILLIAMPS. SINCE BETA IS HIGH IN THE 2N3405, THIS
*    CURRENT IS ALSO THE CURRENT IN THE COLLECTOR
*    HENCE THE CURRENT ACROSS THE CELL.
*.....................................................
```

```
         L=1
       DO 3 N=2,512,2
          Y(L)=(Z(N)*CONST)/RESISTOR
          L=L+1
3      CONTINUE

*.....................................................
*      REMOVE REPETITIVE DATA SHOULD IT OCCUR
*      PLOT NOTHING TO THE LEFT OF X=0.0.
*      PLOT NOTHING BELOW Y=0.0.
*.....................................................

       DO 17 N=1,256
       IF (N.EQ.256) GOTO 40
       IF ((Y(N).EQ.0.0).AND.(Y(N+1).EQ.0.0)) GOTO 17
       IF (((X(N).NE.0.).AND.(X(N).EQ.X(N+1)))
      1.AND.(Y(N).NE.0.)) GOTO 17
       IF ((X(N).NE.0.).AND.(Y(N).EQ.Y(N+1))) GOTO 17

       IF (N.EQ.1) GOTO 70

*.....................................................
*      WRITE RESULTS TO FILE FOR SUBSEQUENT PLOTTING.
*      ALSO WRITE THE DATA TO THE SCREEN.
*.....................................................

70     WRITE (6,'(F6.2,4X,F6.2)') X(N),Y(N)
       WRITE (*,'(I4,4X,F6.2,4X,F6.2)') N,.X(N),Y(N)

*.....................................................
*      WHEN THE VOLTAGE REACHES ZERO, NO FURTHER POINTS
*      ARE REQUIRED TO BE PLOTTED.
*.....................................................

       IF (X(N).EQ.0.0) GOTO 40
17     CONTINUE

40     STOP
       END
```

APPENDIX H.

I. MC68701 INTERNAL EPROM PROGRAMMER BOARD

The following three pages contain the printed circuit board layouts for the microcomputer internal EPROM programmer. Figure 21 is the component side of the printed circuit board. Figure 22 is the solder side of the same board. Figure 23 is the padmaster showing the required component layout. Figures 21 and 22 are photo-ready and suitable for the manufacture of additional boards.

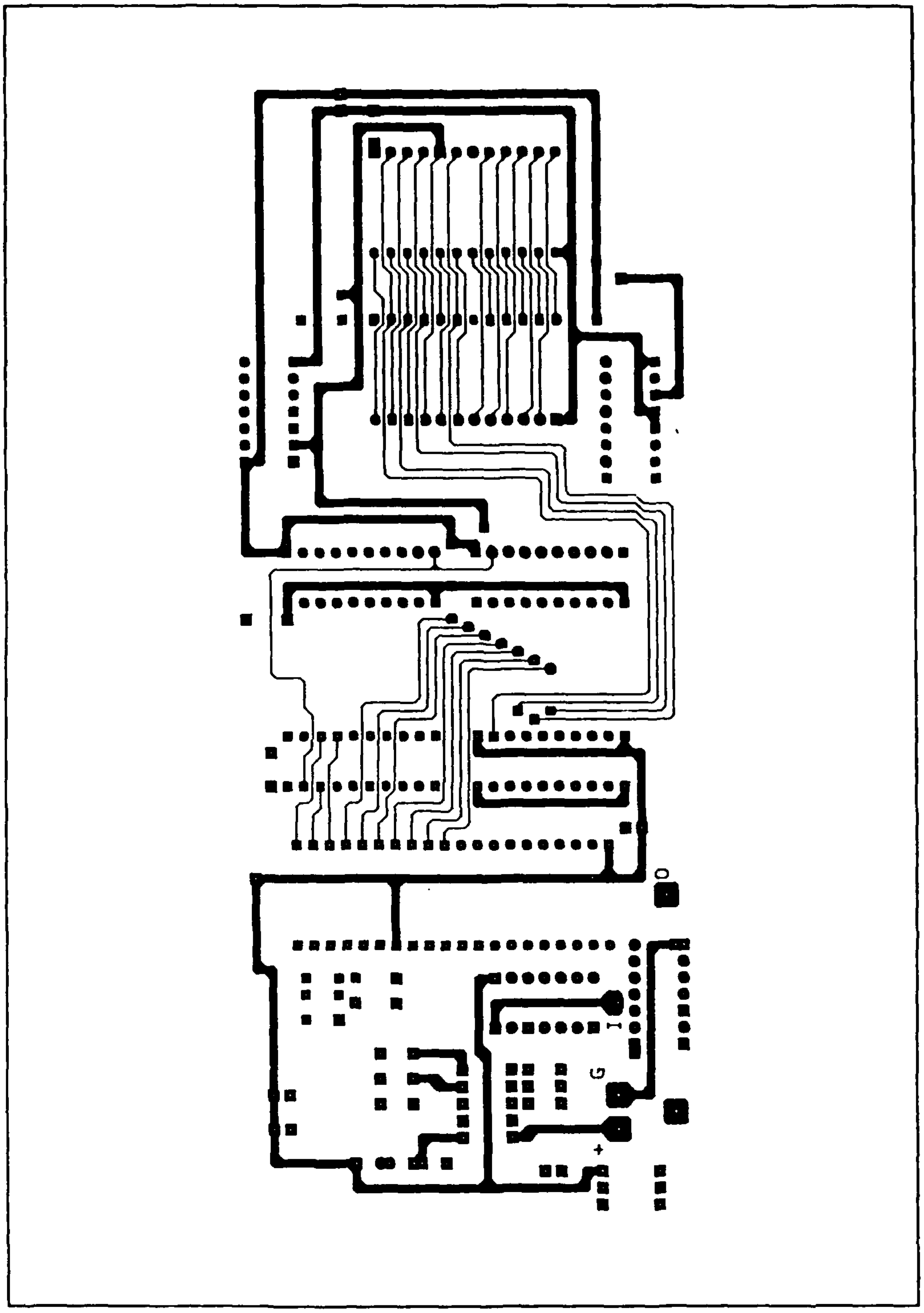


Figure 21 MC68701 Programmer Board
(Component Side)

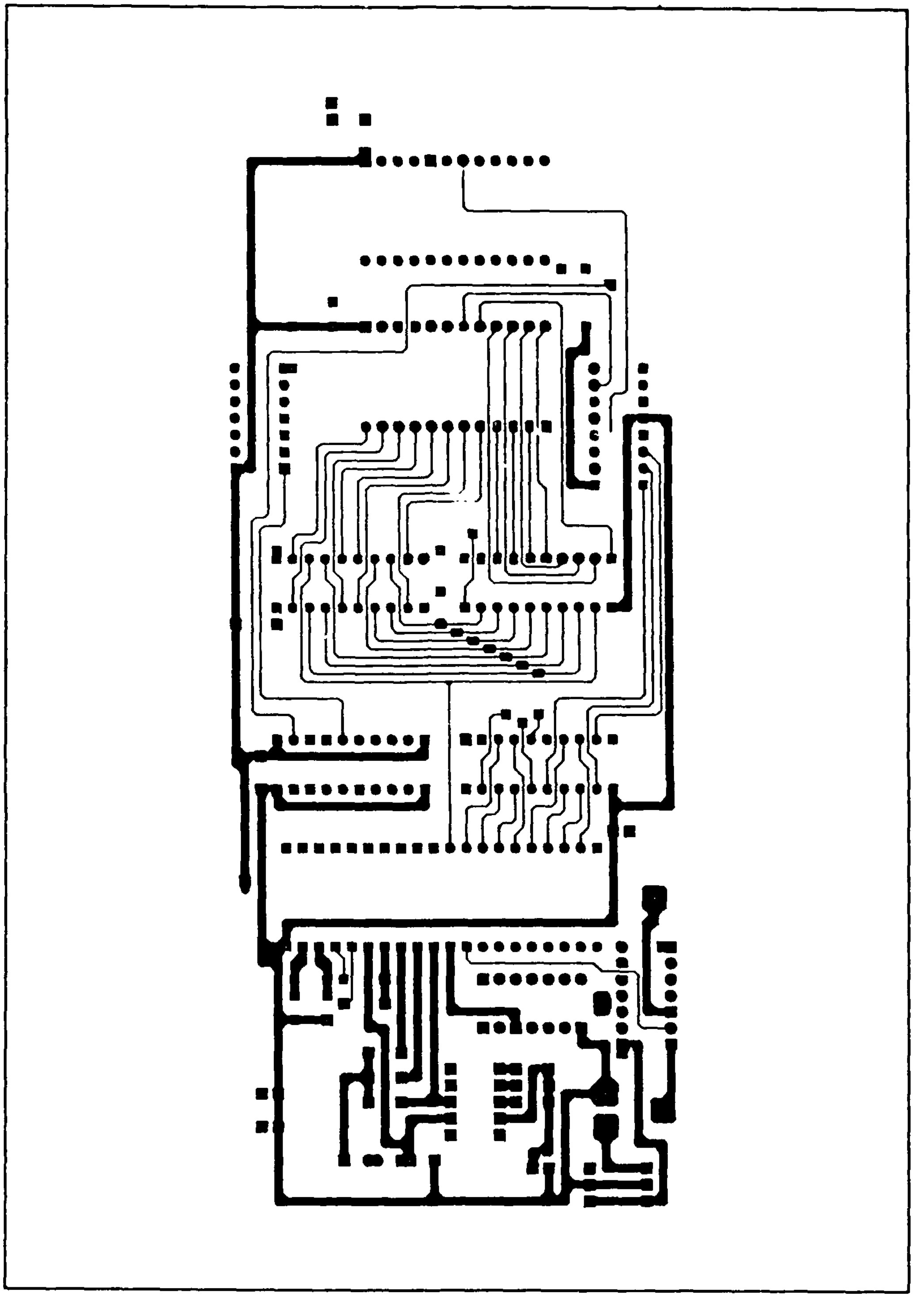


Figure 22 MC68701 Programmer Board
(Solder Side)

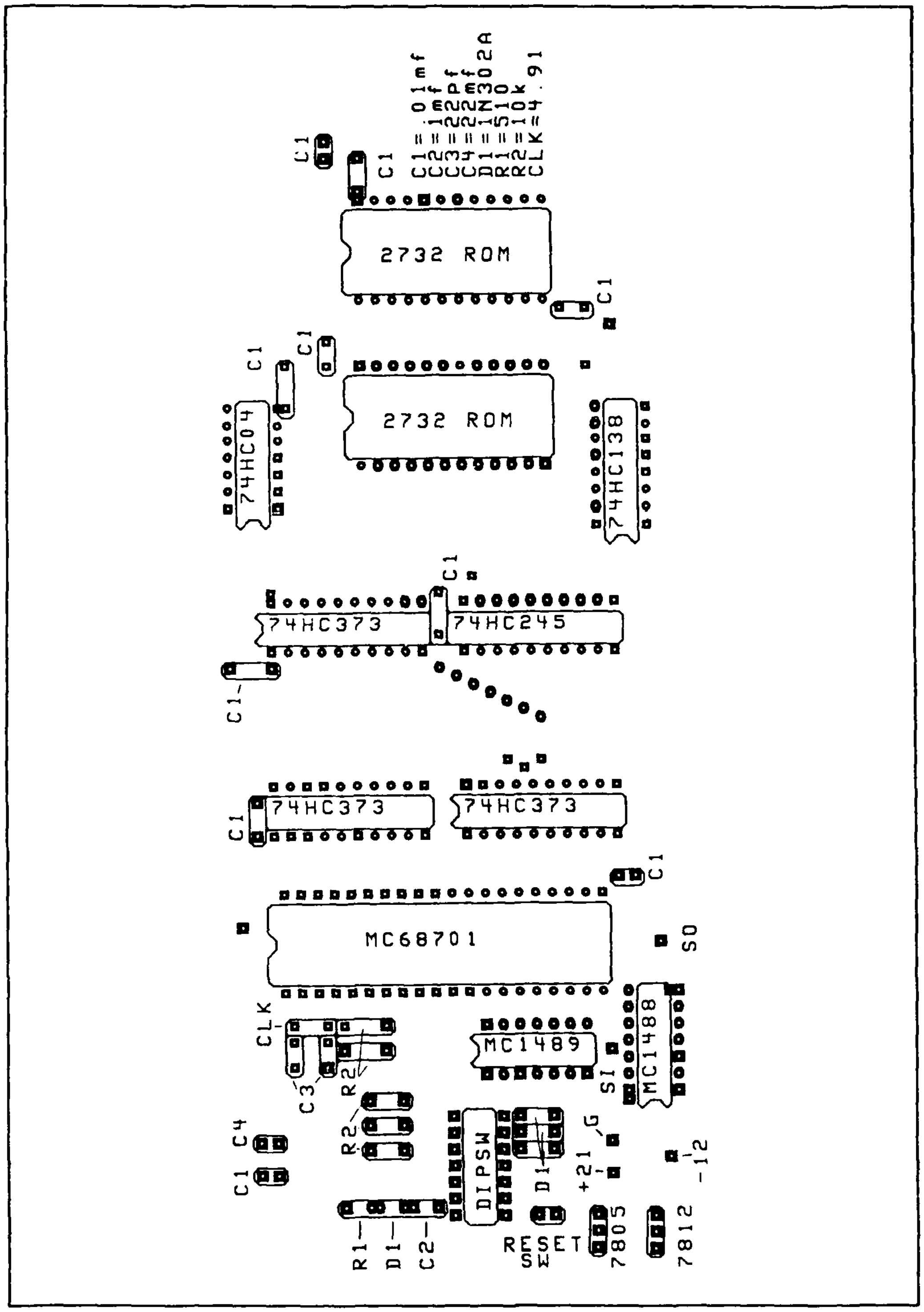


Figure 23 MC68701 Programmer Card
(Component Layout)

APPENDIX I.

I. MAIN COMPUTER BOARD

The following pages contain the printed circuit board layouts for the satellite experiment main computer. Figure 24 is the component side of the board. Figure 25 is the solder side of the board. Figure 26 is the padmaster showing the required component layout. Figures 24 and 25 are photo-ready and suitable for the manufacture of additional boards.

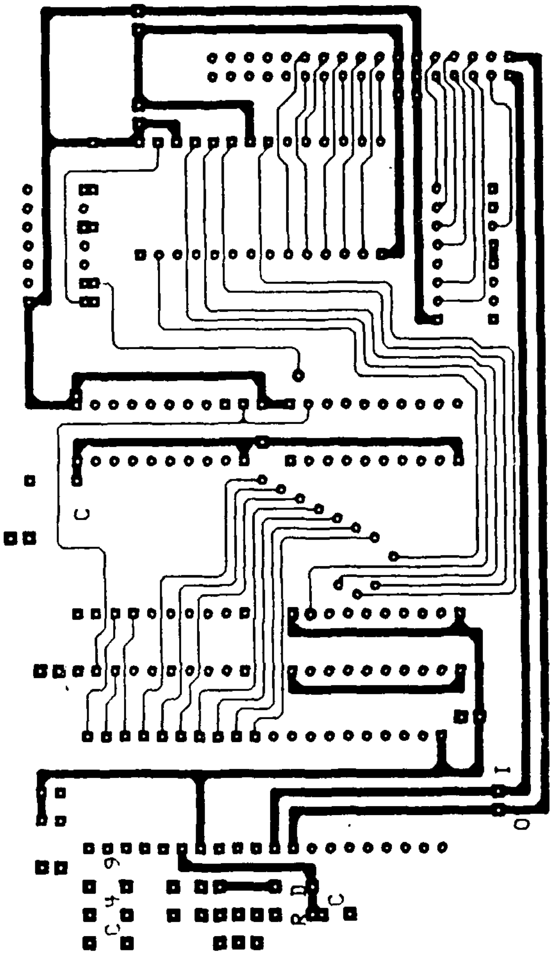


Figure 24 Main Computer Board
(Component Side)

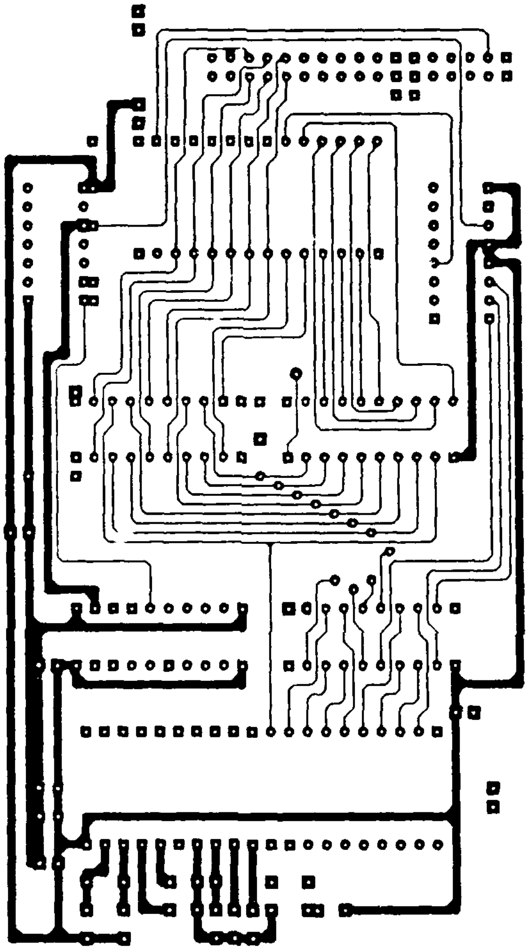


Figure 25 Main Computer Board
(Solder Side)

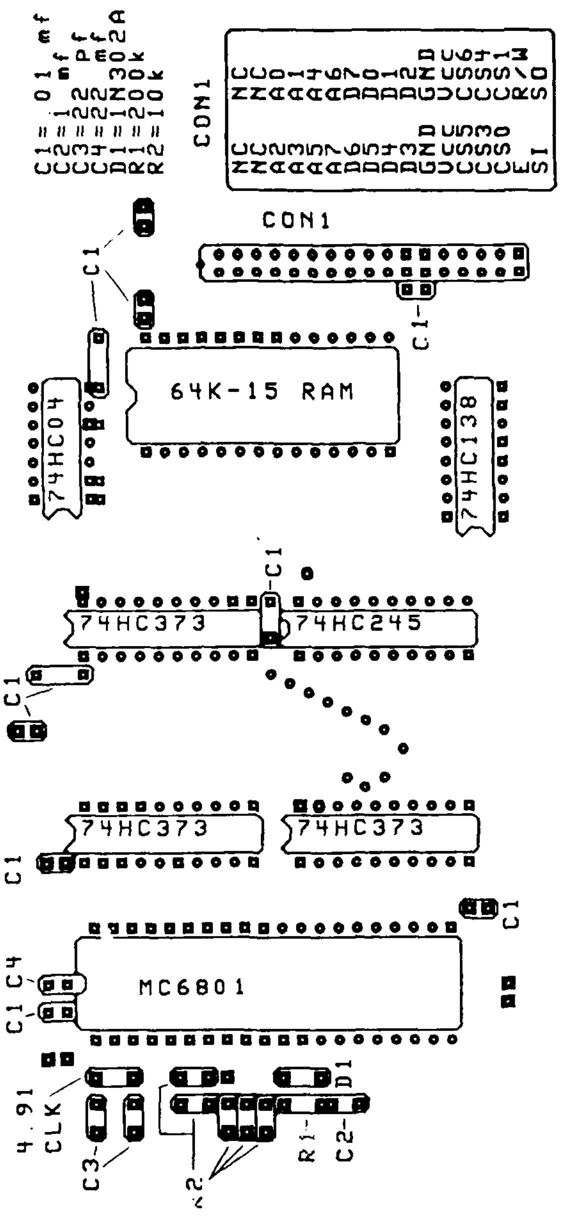


Figure 26 Main Computer Board
(Component Layout)

APPENDIX J.

I. DATA ACQUISITION BOARD

The following pages contain the printed circuit board layouts for the satellite experiment data acquisition board. Figure 27 is the component side of the board. Figure 28 is the solder side of the board. Figure 29 is the padmaster showing the required component layout. Figures 27 and 28 are photo-ready and suitable for the manufacture of additional boards.

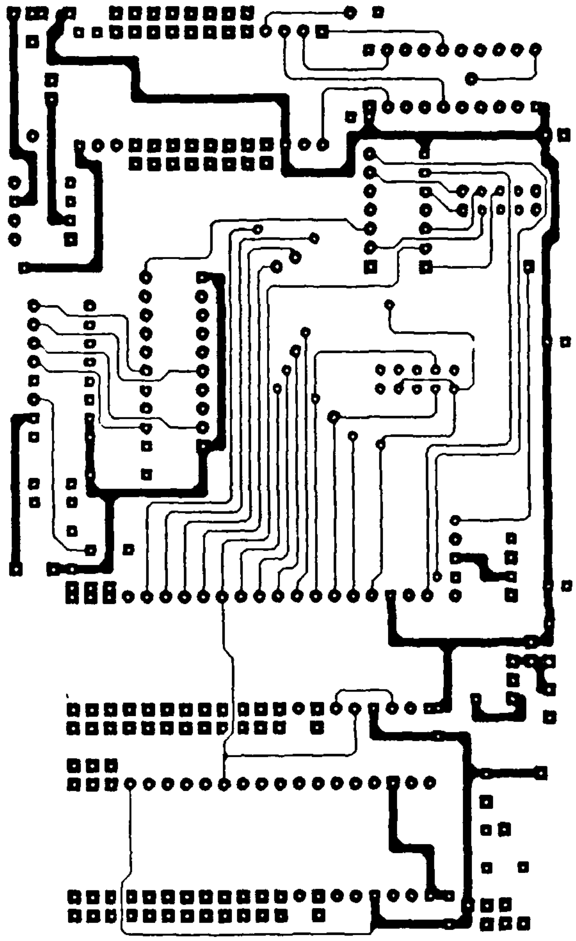


Figure 27 Data Acquisition Board
(Component Side)

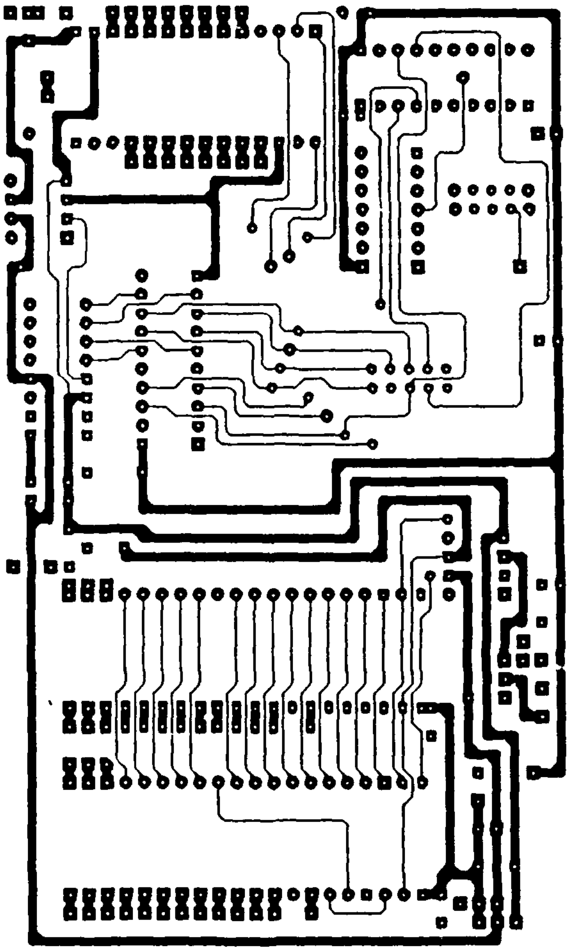


Figure 28 Data Acquisition Board
(Solder Side)

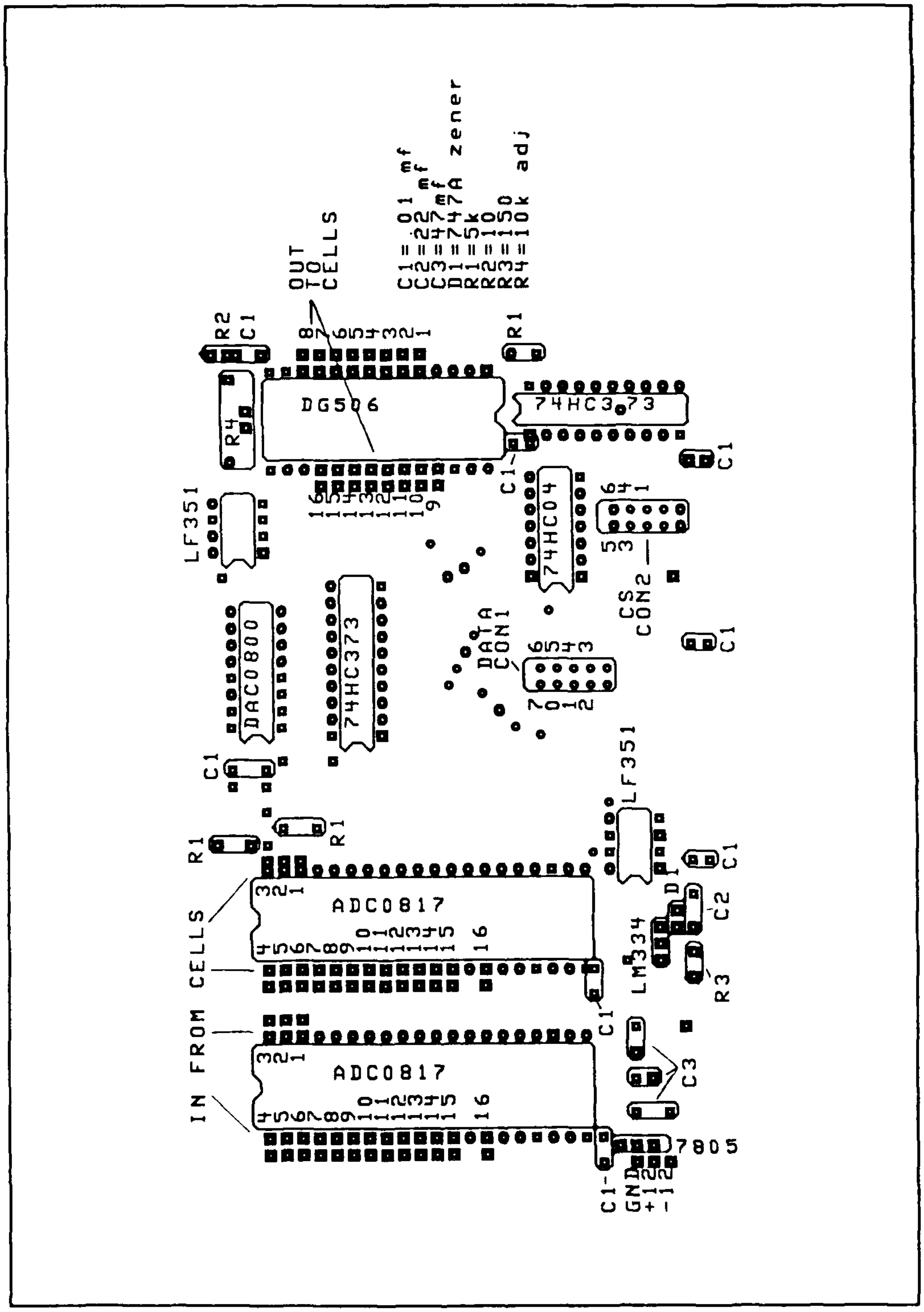


Figure 29 Data Acquisition Board
(Component Layout)

APPENDIX K.

I. ANALOG INTERFACE BOARD

The following pages contain the printed circuit board layouts for the satellite experiment analog interface board. Figure 30 is the component side of the board. Figure 31 is the solder side of the board. Figure 32 is the padmaster showing the required component layout. Figures 30 and 31 are photo-ready and suitable for the manufacture of additional boards.

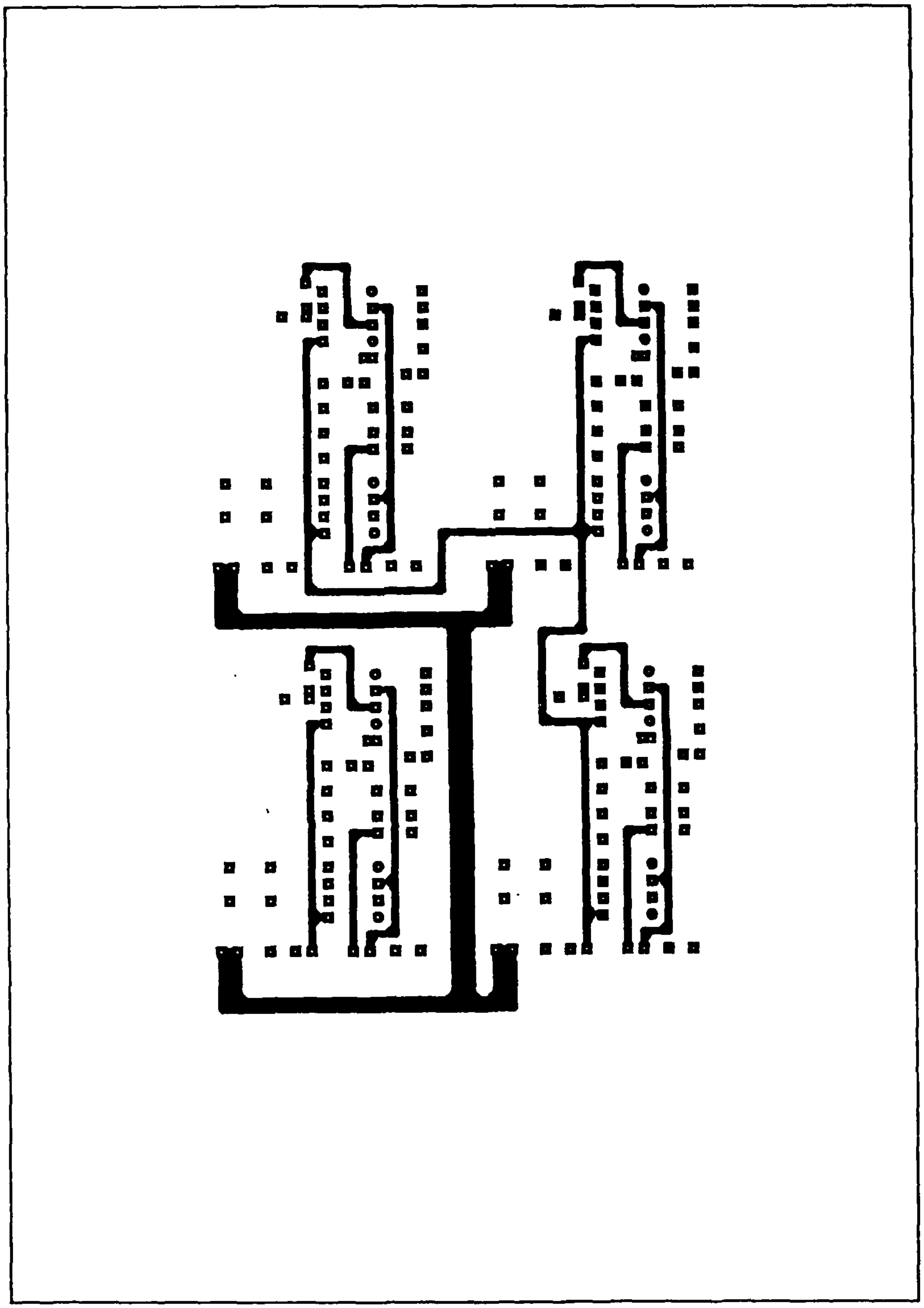


Figure 30 Analog Interface Board (Component Side)

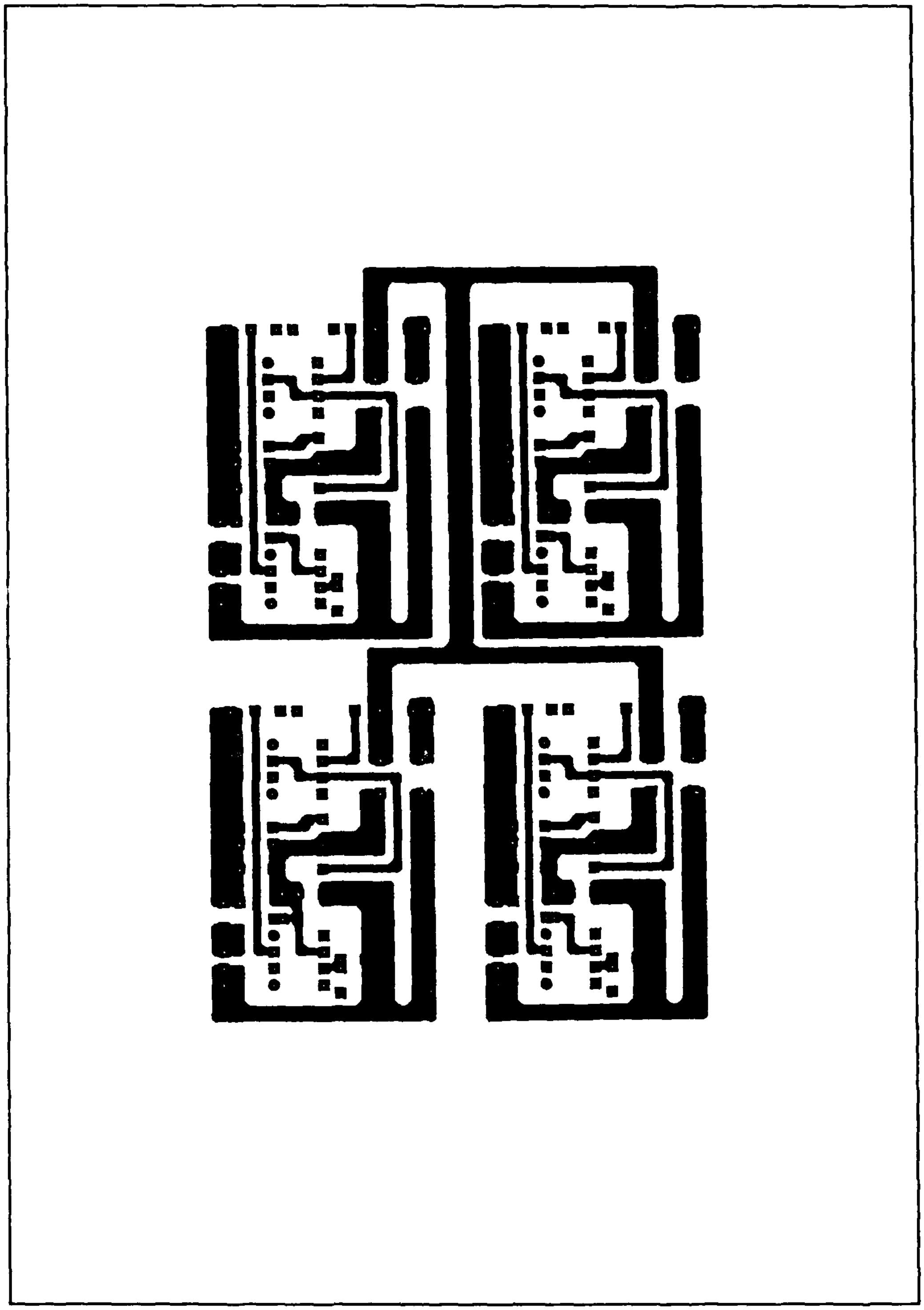


Figure 31 Analog Interface Board
(Solder Side)

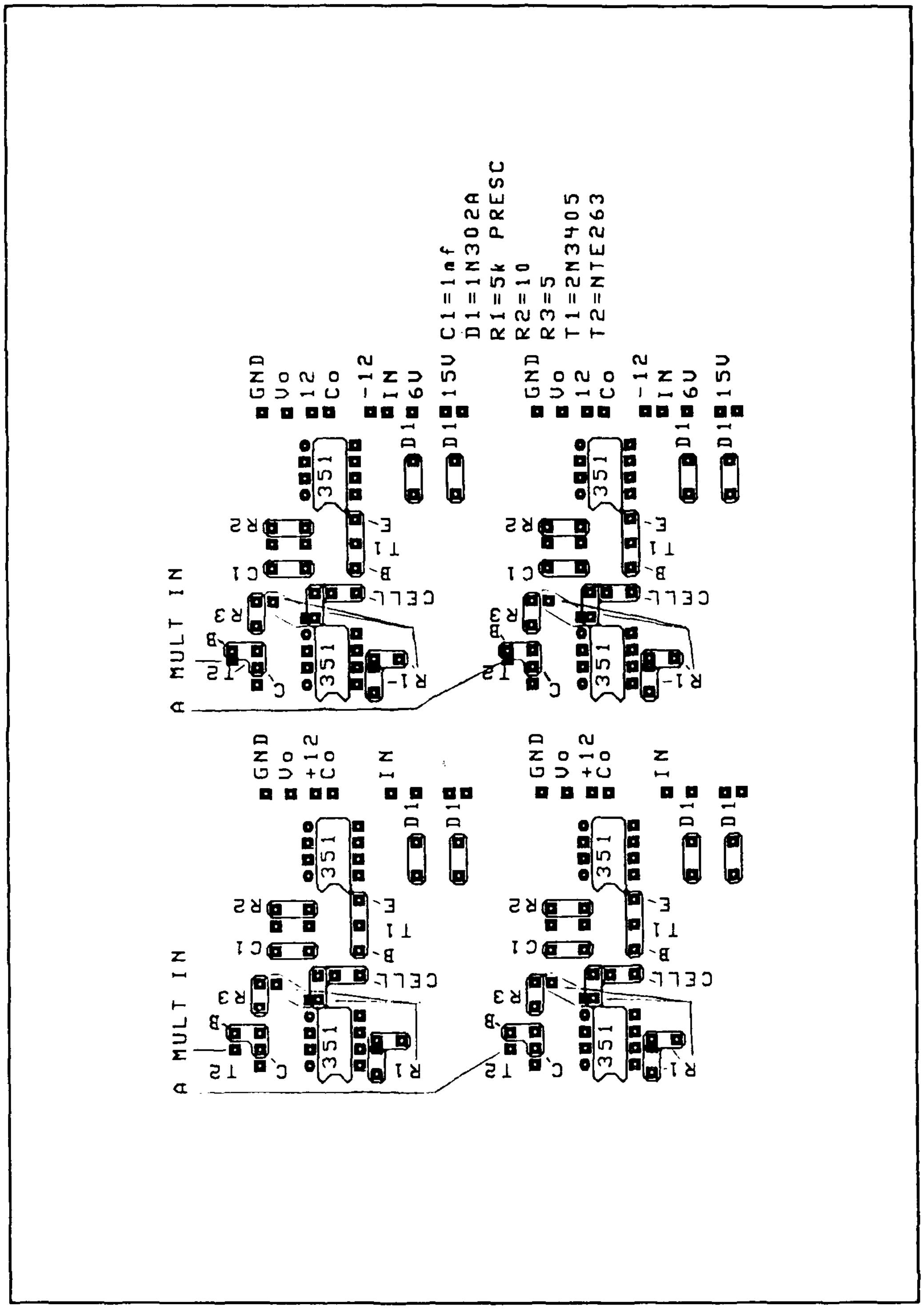


Figure 32 Analog Interface Board
(Component Layout)

APPENDIX L.

I. ANNEAL INTERFACE BOARD

The following pages contain the printed circuit board layouts for the satellite experiment anneal interface board. Figure 33 is the component side of the board. Figure 34 is the solder side of the board. Figure 35 is the padmaster showing the required component layout. Figures 33 and 34 are photo-ready and suitable for the manufacture of additional boards.

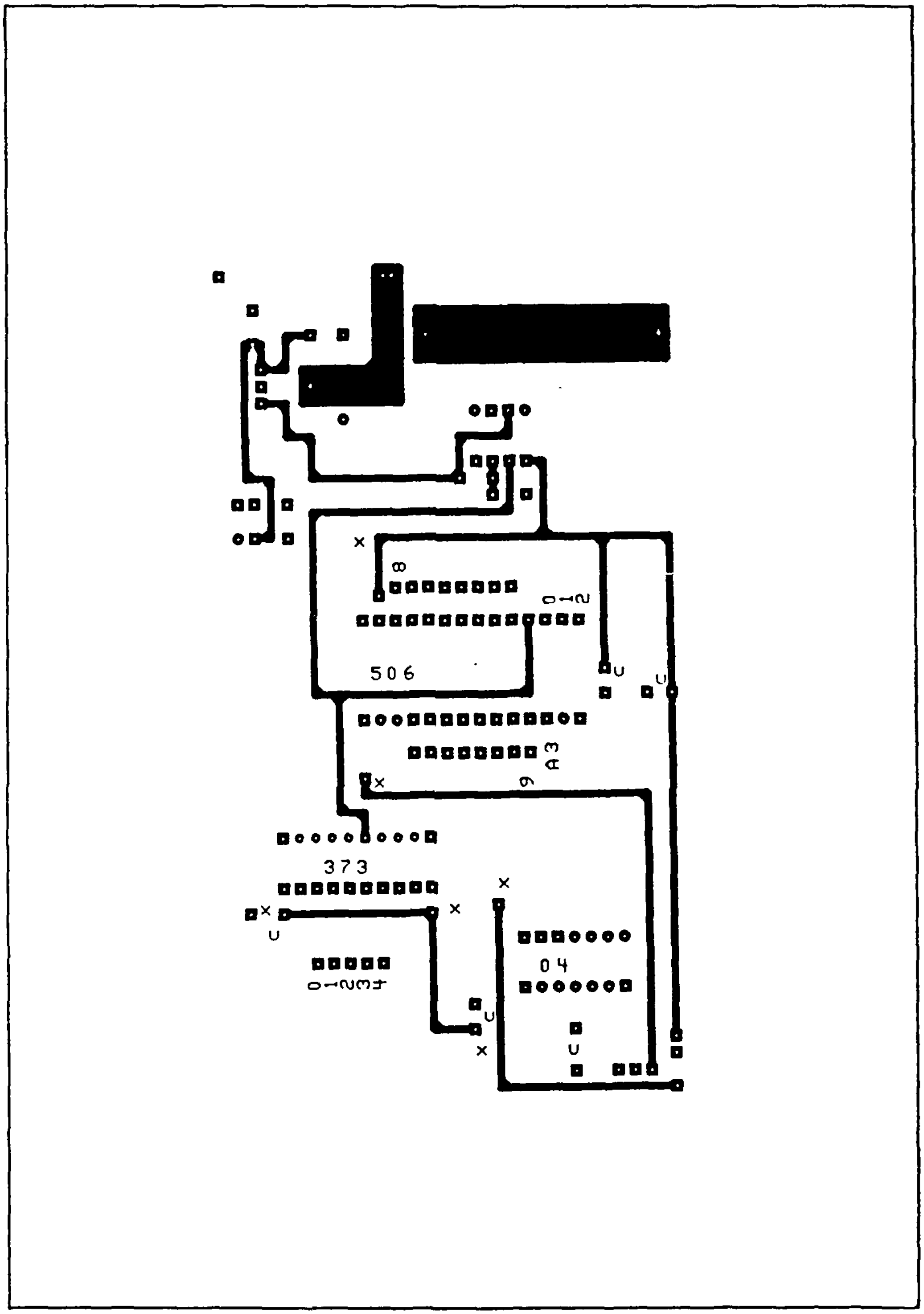


Figure 33 Anneal Interface Board
(Component Side)

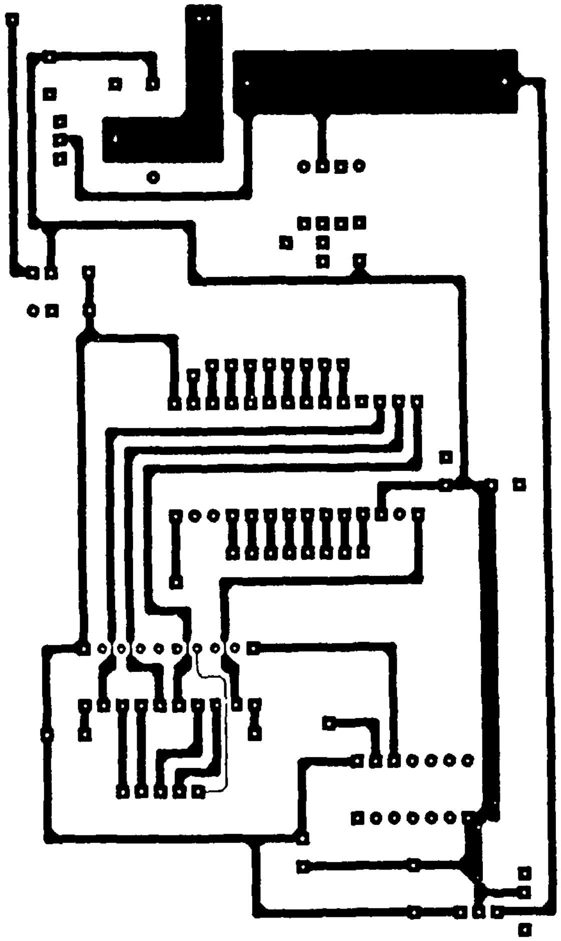


Figure 34 Anneal Interface Board
(Solder Side)

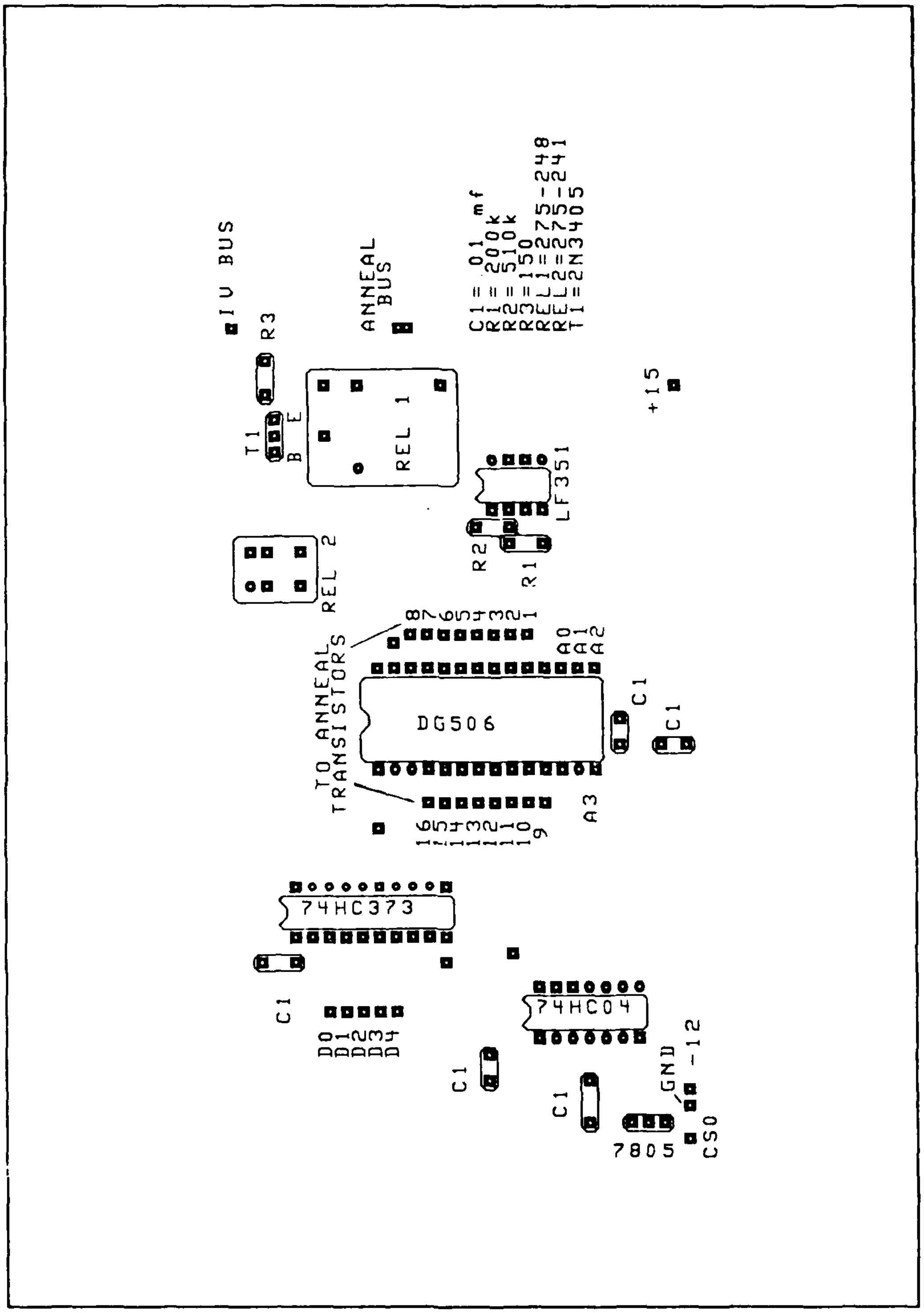


Figure 35 Anneal Interface Board
(Component Layout)

APPENDIX M.

I. RS232 INTERFACE BOARD

The following pages contain the printed circuit board layouts for the satellite experiment RS232 interface board. Figure 36 is the component side of the board. Figure 37 is the solder side of the board. Figure 38 is the padmaster showing the required component layout. Figures 36 and 37 are photo-ready and suitable for the manufacture of additional boards.

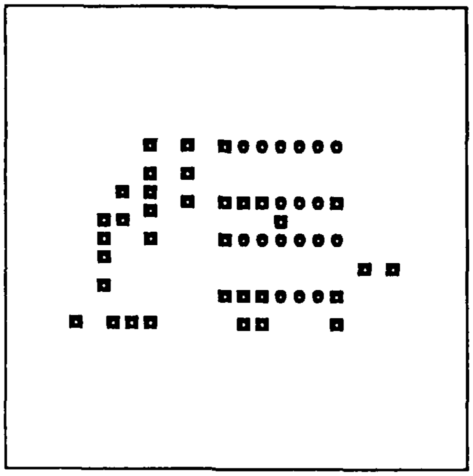


Figure 36 RS232 Interface
(Component Side)

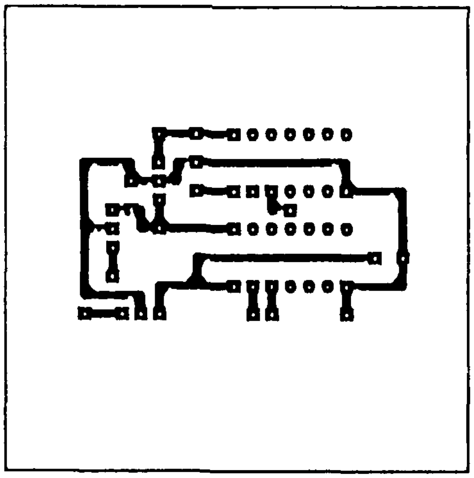


Figure 37 RS232 Interface
(Solder Side)

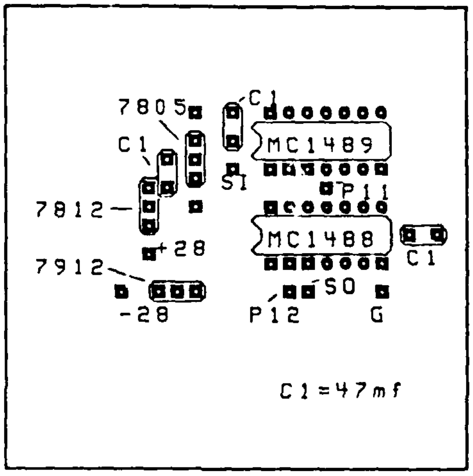


Figure 38 RS232 Interface
(Component Layout)

LIST OF REFERENCES

1. Wise, J. F., ***Solar Power Requirements for Military Space Vehicles***, Seventeenth IEEE Photovoltaic Specialists Proceedings, pp. 17-22, May 1984.

2. Callaway, R. K., ***An Autonomous Circuit for the Measurement of Photovoltaic Devices Parameters***, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1986.

3. Horne, W. E., Day A. C. and Thompson, M. A., ***Burst Annealing of Proton Damage In Silicon Solar Cells***, Seventeenth IEEE Photovoltaic Specialists Conference, pp. 1097-1102, May 1984.

4. Yamaguchi, M., and Amano, C., ***Cobalt 60 Gamma-Ray and Electron Irradiation Damage of GaAs Single Crystals and Solar Cells***, Journal of Applied Physics, Vol 54, n. 9, pp. 5021-5029, September 1983.

5. Clark, F. C., ***An Experimental Test of Minority Carrier Annealing of Gallium Arsenide Solar Cells Using Forward-Biased Current***, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1986.

6. Severns, J. G., and others, ***Experiment Integration on Spacecraft LIPS III***, Second AIAA/USU Small Satellite Proceedings, September 1988.

7. Motorola Corporation, ***MC6801 8-Bit Single Chip Microcomputer Reference Manual***, MC6801RM(AD2), 1983

BIBLIOGRAPHY

Bishop, R., ***Basic Microprocessors and the 6800***, Hayden Book Company, Inc.,1979.

Clements, A., ***Microprocessor Systems Design***, PWS Publishers, 1987.

Francis, R. W., and Betz, F., ***Two Years of On-Orbit Gallium Arsenide Performance from the LIPS Solar Cell Panel Experiment***, Proceeding of Space Photovoltaic Research and Technology Conference, pp. 1-12, May 1985.

Gold, D., ***High Energy Electron Radiation Degradation of Gallium Arsenide Solar Cells***, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1986.

Jet Propulsion Laboratory, Publication 43-38, ***Solar Cell Array Design Handbook***, Volume 1, California Institute of Technology, Pasadena, California, 1976.

Jet Propulsion Laboratory, Publication 82-69, ***Solar Cell Radiation Handbook***, California Institute of Technology, Pasadena, California, 1982.

Leventhal, L., ***6800 Assembly Language Programming***, Osborne/McGraw-Hill, 1978.

Motorola Corporation, ***M6800 Microprocessor Applications Manual***, 1975.

Sommers, R., and Michael S., ***On-Orbit Annealing of Satellite Solar Panels***, Second AIAA/USU Small Satellite Proceedings, September 1988.

Staats, R. L., ***Forward-Biased Current Annealing of Radiation Damaged Gallium Arsenide and Silicon Solar Cells***, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1987.

Trumble, T. M., and Betz, F., ***Evaluation of Gallium Arsenide Solar Panel on the LIPS II Satellite***, 17th IEEE Photovoltaic Specialists Proceedings, pp. 1108-1113, May 1984.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Commander Naval Space Command Attn: Code N155 Dahlgren, VA 22448	1
4.	United States Space Command Attn: Technical Library Peterson AFB, CO 80914	1
5.	Director Navy Space Systems Division (OP-943) Washington, DC 20350-2000	1
6.	Professor J. Powers, Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, Ca 93943	1
7.	Professor R. Panholzer Code 62Pz Naval Postgraduate School Monterey, CA 93943	1
8.	Professor S. Michael Code 62Mi Naval Postgraduate School Monterey, CA 93943	4

9.	Professor H. Loomis Code 62Lm Naval Postgraduate School Monterey, CA 93943	2